Introduction de raisonnement probabiliste dans la méthode B événementiel

Mohamed Amine Aouadhi

UNIVERSITE BRETAGNE LOIRE

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Mohamed Amine AOUADHI

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université de Nantes
sous le sceau de l'Université Bretagne Loire

École doctorale : Sciences et Technologies de l'Information, et Mathématiques

Discipline : Informatique, section CNU 27
Unité de recherche : Laboratoire des Sciences du numérique de Nantes (LS2N)

Soutenue le 29 septembre 2017

Introduction de raisonnement probabiliste dans la méthode B événementiel

JURY

Président :	**M. Benoît CAILLAUD**, Directeur de recherche, INRIA Rennes / IRISA
Rapporteurs :	**M. Dominique MÉRY**, Professeur des Universités, LORIA, Université de Nancy
	M^me^ Kouchnarenko OLGA, Professeur des Universités, Université de Franche-comté
	* * *
Directeur de thèse :	**M. Claude JARD**, Professeur des Universités, LS2N, Université de Nantes
Co-encadrants de thèse :	**M. Arnaud LANOIX**, Maitre de conférences, LS2N, Université de Nantes
	M. Benoît DELAHAYE, Maitre de conférences, LS2N, Université de Nantes

Remerciements

Cette thèse a été menée au sein de l'équipe AeLoS du laboratoire LS2N de l'Université de Nantes.

J'aimerais tout d'abord remercier M. Benoît Caillaud, d'avoir accepter de présider le jury de cette thèse. Je remercie également M. Dominique Mery et Mme. Olga Kouchnarenko d'avoir pris le temps de lire en détails mes travaux, de les corriger et de les commenter. Je remercie également tous les membres du jury pour m'avoir fait l'honneur de leur participation.

Je remercie mon directeur de thèse M. Claude Jard pour avoir donné une direction à mon travail, et ce, avec la plus grande rigueur et gentillesse.

Je remercie mes encadrants de thèse, Benoît Delahaye et Arnaud Lanoix, pour l'encadrement, le soutien et la disponibilité. Leurs conseils, leurs suggestions de lecture, leurs commentaires, leurs corrections et leurs qualités scientifiques ont été très précieux pour mener à bien ce travail. Je sais que tout ce que vous m'avez demandé était dans mon intérêt et je vous en remercie énormément.

Je remercie chaleureusement tous les membres de l'équipe AeLoS, permanents ou temporaires, encore présents ou partis vers d'autres horizons. Je remercie en particulier M. Christian Attiogbé pour le soutien qu'il m'a accordé.

Finalement, je remercie profondément ma famille et mes ami(e)s de m'avoir toujours soutenu. Un remerciement particulier pour ma mère, Ajmia, qui a toujours cru en moi et m'a toujours encouragé de aller de l'avant.

Table des matières

1 Introduction **9**

2 B événementiel et systèmes de transitions **17**
- 2.1 Introduction . . . 17
- 2.2 Modèle B événementiel . . . 18
  - 2.2.1 Contexte . . . 18
  - 2.2.2 Machine . . . 19
  - 2.2.3 Événement . . . 20
  - 2.2.4 Substitution généralisée . . . 21
- 2.3 Cohérence d'un modèle B évenementiel . . . 22
  - 2.3.1 Correction des expressions . . . 22
  - 2.3.2 Obligations de preuve de machine . . . 23
- 2.4 Raffinement . . . 24
  - 2.4.1 Raffinement en B événementiel . . . 24
  - 2.4.2 Exactitude du raffinement . . . 26
  - 2.4.3 Convergence des nouveaux événements . . . 27
- 2.5 Étude de cas : Protocole Pair à Pair . . . 28
  - 2.5.1 Stratégie du développement du protocole en B événementiel . . . 28
  - 2.5.2 La première machine . . . 29
  - 2.5.3 Deuxième machine . . . 29
  - 2.5.4 Troisième machine . . . 32
- 2.6 Systèmes de transitions . . . 33
  - 2.6.1 Systèmes de transitions . . . 33
  - 2.6.2 Systèmes de transitions et probabilités . . . 35
  - 2.6.3 Sémantique opérationnelle d'une machine B événementiel . . . 37
- 2.7 B Événementiel et probabilités . . . 38

3 B événementiel probabiliste **43**
- 3.1 Introduction . . . 43
- 3.2 Introduction des probabilités en B événementiel . . . 44
  - 3.2.1 Non-déterminisme en B événementiel : Exemple . . . 44
  - 3.2.2 Introduction des probabilités dans la machine Mch . . . 46
- 3.3 B événementiel probabiliste . . . 48
  - 3.3.1 Choix de l'événement à exécuter . . . 48
  - 3.3.2 Choix uniforme des valeurs des paramètres . . . 49
  - 3.3.3 Les substitutions probabilistes . . . 49
  - 3.3.4 Machine B événementiel probabiliste . . . 51
  - 3.3.5 Étude de cas : protocole pair à pair en B événementiel probabiliste . . . 53
- 3.4 Cohérence d'une machine B évenementiel probabiliste . . . 54
  - 3.4.1 Nouvelles obligations de preuve . . . 54

3.4.2 Adaptation des obligations de preuve standard . . . 56
3.4.3 Étude de cas : obligations de preuve appliquées au protocole pair à pair . . 57
3.5 Sémantique opérationnelle d'une machine B événementiel probabiliste . . . 57
3.5.1 Notations . . . 57
3.5.2 Construction du système de transitions probabiliste correspondant à une machine probabiliste . . . 59
3.5.3 Sémantique opérationnelle exprimée en termes de chaîne de Markov discrète 61
3.5.4 Étude de cas : sémantique opérationnelle de la machine P2P$_p$ du protocole pair à pair . . . 62

4 B Événementiel mixte **67**
4.1 Introduction . . . 67
4.2 Machine B événementiel mixte . . . 68
4.2.1 Description . . . 68
4.3 Cohérence d'une machine B événementiel mixte . . . 69
4.3.1 Cas général . . . 69
4.3.2 Cas des machines mixtes partielles . . . 70
4.4 Sémantique opérationnelle d'une machine B événementiel mixte . . . 71
4.4.1 Construction de la sémantique opérationnelle d'une machine B événementiel mixte . . . 71
4.4.2 Exemples d'illustration . . . 72
4.4.3 Sémantique opérationnelle . . . 74
4.5 Etude de cas : protocole pair à pair . . . 76

5 Raffinements probabilistes **81**
5.1 Introduction . . . 81
5.2 Probabilisation . . . 83
5.2.1 Obligations de preuve de faisabilité de la probabilisation . . . 84
5.2.2 Automatisation du processus de probabilisation . . . 85
5.2.3 Étude de cas . . . 85
5.3 Introduction de nouveaux événements par raffinement . . . 88
5.3.1 Convergence presque certaine en B événementiel dans la littérature . . . 89
5.3.2 Convergence presque certaine dans notre proposition de B événementiel probabiliste . . . 89
5.3.3 Étude de cas : convergence presque certaine dans le protocole P2P . . . 100
5.4 Généralisation de l'introduction d'événements par raffinement . . . 105
5.4.1 Ajout de nouveaux événements probabilistes au sein d'une machine B événementiel standard ou mixte . . . 107
5.4.2 Ajout d'événements standard dans une machine B événementiel mixte ou probabiliste . . . 107
5.4.3 Ajout simultané d'événements standard et probabilistes . . . 107

6 Études de cas **111**
6.1 Système d'atterrissage d'un avion . . . 111
6.2 Stratégie de développement du train d'atterrissage en B événementiel . . . 112
6.2.1 La première machine . . . 113
6.2.2 La deuxième machine . . . 115
6.3 Système de freinage . . . 117
6.3.1 Stratégie du développement du système de freinage en B événementiel . . . 118
6.3.2 La première machine . . . 118
6.3.3 La deuxième machine . . . 120

6.3.4 La troisième machine : version probabiliste 121
6.3.5 La quatrième machine 122
6.3.6 La cinquième machine 122

7 Extension de la plateforme Rodin **125**
7.1 Introduction 125
7.2 La plateforme Rodin 125
7.2.1 Architecture de Rodin 125
7.2.2 Processus de développement d'un modèle B événementiel dans Rodin 127
7.3 Intégration de notre extension probabiliste dans Rodin 129
7.3.1 Description de l'extension 129

8 Conclusion **135**

1

Introduction

Si nous observons notre environnement, nous pouvons constater qu'une multitude de systèmes informatiques nous entourent. Ces systèmes sont devenus au cours de ces dernières années de plus en plus complexes, requérant une pluridisciplinarité très large pour les concevoir. Ils représentent également des enjeux stratégiques (économiques, sociétaux) importants, puisqu'ils interviennent dans des composantes critiques de systèmes aussi divers que l'aérospatial, l'aéronautique, les systèmes nucléaires, la télécommunication, le transport terrestre, la santé, le commerce électronique... Intervenant dans des domaines stratégiques, ces systèmes informatiques doivent être vérifiés, la moindre défaillance pouvant coûter très cher et causer des pertes irrémédiables : pertes d'informations, pertes financières ou même pertes humaines. Nous distinguons deux types de dysfonctionnement des systèmes. Le premier type concerne les problèmes matériels, qui se produisent lorsqu'un système connait une panne physique. Le deuxième type concerne les problèmes logiciels, appelés *erreurs* par abus de langage, qui surviennent lorsqu'un composant ne réalise pas précisément les fonctionnalités pour lesquels il a été conçu. Parmi les erreurs qui ont marqué l'histoire, nous pouvons citer par exemple celle de la fusée Ariane 5 [34] (700 millions de dollars de perdus) ou celle du missile Patriot [43] (28 morts).

La production de systèmes informatiques *sûrs* est ainsi une activité complexe qu'il est difficile de mettre en oeuvre. La garantie de fonctionnement de ces systèmes est un enjeu crucial lors de leur développement et qui doit être pris en considération dès les premières étapes du développement. Face à ce besoin croissant et nécessaire de produire des systèmes informatiques sûrs, de nombreuses solutions de développement ont été proposées.

La plupart des méthodes de développement classiques ne permettent pas de décrire d'une manière concise et correcte le comportement des systèmes. La limitation principale concerne l'activité de vérification qui n'est pas bien développée dans ces méthodes. Généralement, cette activité s'applique avant la phase d'implantation d'un système. Partant d'une spécification détaillée d'un système exprimée dans un modèle possédant une sémantique formelle, cette activité consiste à vérifier les propriétés comportementales de la spécification afin de détecter et corriger le plus tôt possible les éventuelles erreurs de conception. Notons que plus les erreurs de conception sont détectées tardivement, plus les coûts de corrections sont élevés.

Afin de s'assurer du bon fonctionnement d'un système, une solution est donc d'adopter une démarche formelle de spécification et de vérification.

Contexte scientifique : l'utilisation des méthodes formelles

Les méthodes formelles regroupent tout un ensemble de notations et de concepts mathématiques et logiques permettant une description mathématique du système et une vérification basée sur les éléments de base de ces notations et concepts. Cette démarche consiste à vérifier si un modèle mathématique regroupant les comportements possibles du système satisfait ou non les propriétés à vérifier (souvent exprimées sous forme de fomules de logiques mathématiques). Les comportements du système sont spécifiés à l'aide d'un langage formel ayant une sémantique précise souvent basée sur des systèmes de transitions. Cette description doit éliminer toute ambiguité existante au niveau de l'expression de ces comportements en langage naturel. La vérification formelle doit montrer que tous les comportements du système satisfont les propriétés désirées.

L'utilisation des méthodes formelles dans une démarche de développement peut se diviser en deux grandes phases : la spécification et la vérification.

La phase de spécification formelle

La spécification formelle est la première étape d'un processus formel de développement. Elle consiste à donner une description formelle d'un système et de ses propriétés en utilisant un langage ayant une syntaxe et une sémantique définies mathématiquement. Cette description formelle permet de lever toutes les ambiguités rencontrées dans la description donnée en langage naturel et de fournir ainsi une description rigoureuse et claire du système et des propriétés à vérifier.

Une pléthore de méthodes de spécification formelle existent. Ces méthodes sont adaptées à plusieurs types de systèmes informatiques (systèmes distribués, systèmes synchrones, systèmes concurents, etc.). Deux grandes familles de formalismes de modélisation se distinguent : les formalismes à base de preuve tels que la méthode B [16], la méthode B événementiel [17], VDM [66], Action Systems [25] ou le langage Z [89] et les formalismes à base de systèmes de transitions tels que les Statecharts [55, 56], TLA [71], etc. Chacune de ces familles offre des avantages et des inconvénients spécifiques. Dans ce document, nous considérons un formalisme à base de preuve, le B événementiel.

Pour la modélisation des systèmes réels (matériels, logiciels ou encore naturels), il est nécessaire de prendre en compte des aspects quantitatifs tels que le temps, les ressources énergétiques, les incertitudes comportementales, la qualité des matériels, etc. Il devient ainsi compliqué de modéliser ces systèmes de manière exacte et discrète et les méthodes de spécification présentées précédemment ne permettent pas la prise en compte de ces aspects. Une solution est d'utiliser la théorie des probabilités pour modéliser les aspects "incertains" de ces systèmes : les incertitudes sont modélisées par des distributions de probabilité, et on parle ainsi de modèles stochastiques. Au lieu de vérifier des propriétés exactes de manière binaire, on s'intéresse dans ce cas à la vérification formelle des propriétés quantitatives. Cela permet de prendre en considération d'autres propriétés telles que la fiabilité [96], la réactivité [38, 94], le temps de réponse, la disponibilité des systèmes, l'évolution continue, la consommation énergétique, etc.

Afin de modéliser ces aspects probabilistes, des formalismes probabilistes de modélisation ont été développés dans la littérature. La plupart de ces formalismes étendent d'autres formalismes de modélisation standards. En effet, de nombreux travaux [51, 86, 57, 48, 29, 64, 22, 63] ont traité l'introduction du raisonnement probabiliste dans des formalismes de modélisation à base de preuve comme la méthode B, Z, Action Systems, etc. Dans [86], les auteurs ont étudié l'expression des probabilités dans le formalisme *Action Systems* et dans [52], les auteurs ont étudié l'analyse de performance des systèmes dans ce formalisme. Dans [50], Haghighi et ses coauteurs

ont proposé une extension probabiliste au formalisme Z. Ils ont aussi étudié dans [51] la spécification des chaines de Markov dans Z. La méthode B a aussi été l'objet de plusieurs travaux d'extension pour supporter le raisonnement probabiliste [57, 59]. De nombreux travaux de recherche [78, 54, 58, 98, 53, 91, 92, 93] ont également étudié l'intégration des probabilités dans la méthode B événementiel, nous détaillerons ces travaux plus loin dans ce manuscrit. Nous notons également que les formalismes à base de systèmes de transitions ont aussi été étendus pour prendre en compte des aspects probabilistes [90, 84, 84, 90].

La phase de vérification

Nous rappelons que l'objectif principal de l'utilisation des méthodes formelles est la vérification du comportement des systèmes. Après la spécification formelle d'un système, c'est le tour de la vérification qui vient : il faut s'assurer que la spécification est correcte et que les propriétés que le système doit satisfaire, sont toutes bien vérifiées. Étant donné un modèle M et une propriété ϕ, le but de la vérification est de garantir formellement que M respecte cette propriété ($M \models \phi$) ou de fournir un contre-exemple dans le cas contraire. L'objectif de cette phase de vérification est de détecter les erreurs commises durant la phase de développement et les diagnostiquer pour pouvoir les corriger au plus tôt et de préférence avant la mise en production du système. Plusieurs méthodes sont utilisées pour la vérification des propriétés des systèmes, nous présentons dans ce qui suit les deux méthodes les plus utilisées, *le model checking* et *le theorem proving*.

- Le *model checking* [65, 40, 31] ou vérification de modèles est une méthode de vérification qui consiste en l'exploration de l'ensemble des exécutions et des états du modèle. Les propriétés sont exprimées la plupart du temps par des formules des logiques temporelles (LTL,CTL,CTL*,CSL...) [83, 30, 39, 87]. Le *model checking* est l'une des méthodes de vérification les plus outillées. De nombreux vérificateurs de modèles ont été développés dans la littérature pour la mise en pratique de cette méthode [61, 72, 47, 99]. Cependant, les techniques de vérification basées sur le *model checking* représentent un problème fondamental que l'on rencontre pour une grande gamme de système réels : le phénomène de l'explosion combinatoire de l'espace d'états [41], qui rend impossible en pratique la phase de vérification car le modèle à vérifier est trop complexe. Les techniques de vérification par *model checking* ont aussi été développées pour les systèmes probabilistes [33, 28]. De nombreuses techniques de manipulation et de combinaison de modèles telles que l'abstraction [68, 44], qui permet de réduire la taille des modèles ou le raffinement [67] qui permet de leur ajouter des détails ont été développées dans ce contexte.

- Le *Theorem proving* : Les méthodes de preuve ou *Theorem proving* ont été introduites par Hoare [60]. Elles permettent de vérifier si un programme satisfait une spécification. Le problème de vérification (le programme satisfait-il la spécification) est exprimé comme un théorème que l'on cherche à prouver à partir d'un ensemble d'axiomes du modèle. Ces méthodes ont été outillées et plusieurs assistants de preuve existent. Nous citons à titre d'exemples les assistants PVS [81], Coq [62], HOL [49], et Isabelle [82]. Ces assistants permettent l'automatisation d'une partie de la preuve. Contrairement au *model checking*, les techniques de preuve ne souffrent pas de l'explosion combinatoire de l'espace d'états. Néanmoins, une intervention humaine est souvent nécessaire pour guider les assistants de preuve, ce qui rend difficile l'utilisation de ces techniques dans le contexte industriel.

  Les techniques de vérification par preuve ont été étendues pour supporter la vérification des comportements probabilistes. Nous notons à titre d'exemples des travaux traitant l'extension probabiliste des formalismes Z [51], Action Systems [86], B [57], HOL [64], Coq [22]. Plusieurs autres travaux ont traité la vérification d'algorithmes probabilistes par des techniques de preuve dédiées [48, 29, 63].

Problématique

Dans cette thèse, nous nous intéressons à la méthode B événementiel. En particulier, nous traitons l'extension du formalisme B événementiel pour supporter le raisonnement probabiliste. B événementiel est une évolution de la méthode B conçue par Jean Raymond Abrial [17]. C'est une méthode dédiée à la modélisation et à la vérification des systèmes distribués réactifs. Elle a été utilisée pour décrire de nombreux cas d'études industriels [17]. Cette méthode se base sur la logique classique du premier ordre en conjonction avec la théorie des ensembles. Elle permet de décrire la dynamique d'un système par un ensemble d'événements agissant sur des variables du système par l'intermédiaire de substitutions. La cohérence d'un modèle B est définie par un ensemble d'obligations de preuve permettant d'assurer qu'un modèle est correct par construction : il faut principalement montrer que le déroulement d'un événement préserve un certain invariant. Le processus de développement en B événementiel est basé sur le raffinement : les détails du système sont introduits progressivement, étape par étape, où chaque nouveau modèle doit préserver le comportement du modèle précédent. Les avantages du B événementiel sont :

- un langage formel de modélisation de haut niveau bien défini et bien adapté à l'échelle industrielle,
- un processus de développement par raffinement largement documenté, et
- un outil pour la spécification et la vérification de modèles B événementiel : Rodin [18].

Malgré son utilisation dans plusieurs cas d'études industriels [17], plusieurs aspects probabilistes ne peuvent pas être décrits en B événementiel d'une manière simple et exacte. En effet, la version basique du B événementiel n'offre pas la possibilité de modéliser des comportements probabilistes. Ainsi, plusieurs travaux de recherche ont traité l'extension du B événementiel avec des probabilités. Cependant, les extensions existantes à ce jour ne permettent pas la description des comportements des systèmes purement probabilistes d'une manière intuitive et simple.

À notre connaissance, le premier travail traitant de la problématique de l'ajout de probabilités au sein de B événementiel a été mené par Abrial et ses coauteurs [78]. Dans cet article, les auteurs traitent les différentes possibilités pour intégrer des probabilités au sein de B événementiel et suggèrent d'introduire les probabilités comme raffinement du non-déterminisme qui peut apparaitre à plusieurs niveaux dans les modèles B événementiel : entre les événements activés simultanément, pour le choix des valeurs des paramètres d'un événement et dans les substitutions non-déterministes. Ce travail peut être considéré comme un ensemble de consignes ou de bonnes pratiques pour intégrer le raisonnement probabiliste en B événementiel. En particulier, il est suggéré que chaque extension de B événementiel pour supporter le raisonnement probabiliste doit respecter certaines exigences :

- Le B événementiel étendu doit rester simple et compréhensible ;
- L'extension doit être générique permettant ainsi la description d'une large classe de systèmes.

À notre connaissance, les autres travaux ayant déjà traité l'intégration de probabilités dans B événementiel [78, 54, 58, 98, 53, 91, 92, 93] se sont contentés de remplacer les substitutions non-déterministes par des substitutions probabilistes. Les extensions issues de ces travaux conservent ainsi du non-déterminisme qui apparaît dans le choix entre événements activables et dans le choix des valeurs des paramètres d'un événement. Ainsi, il est difficile de modéliser des systèmes purement probabilistes dans ces extensions. En effet, il est alors nécessaire d'effectuer des modifications et des réecritures fastidieuses sur les modèles afin d'assurer qu'aucun aspect non-déterministe n'est encore présent, ce qui complique la tâche de modélisation et donne des modèles B événementiel moins lisibles et moins compréhensibles.

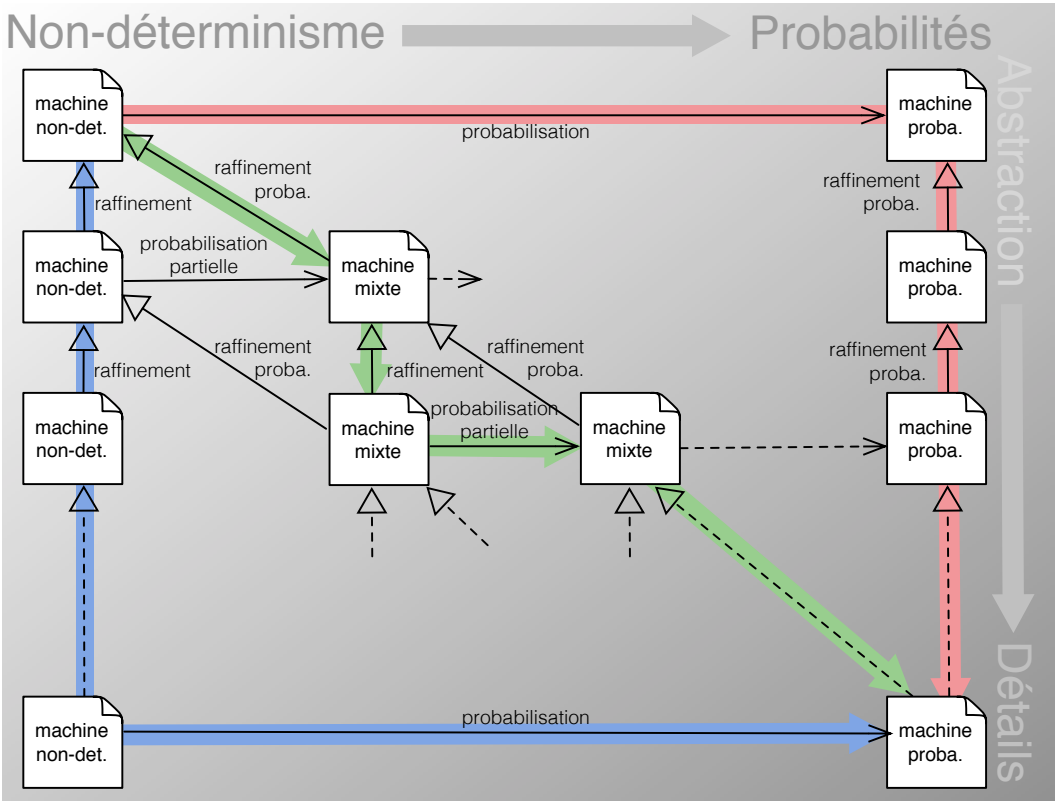

FIGURE 1.1 – Comment introduire des probabilités au sein du B événementiel ?

Ainsi, l'objectif de nos travaux de thèse est de proposer une nouvelle extension probabiliste au B événementiel qui permet l'introduction des probabilités à tous les niveaux où le non-déterminisme peut apparaitre. Une telle extension facilitera donc la description des modèles purement probabilistes et rendant leur écriture plus simple et plus intuitive.

Objectifs de la thèse

L'objectif de cette thèse est d'étudier et de développer, une nouvelle extension probabiliste à la méthode B événementiel. Dans cette extension, nous proposons d'introduire des probabilités à tous les endroits où le non-déterminisme peut apparaître en B événementiel. Contrairement aux autres travaux qui ont remplacé une seule source de non-déterminisme par des probabilités, cette extension facilite la spécification des modèles purement probabilistes en B événementiel. Cependant, dans certains cas le comportement de certains systèmes fait intervenir à la fois du non-déterminisme et des probabilités. Ainsi, on peut souhaiter conserver du non-déterminisme à certains endroits et le remplacer à d'autres endroits, ce qui donne des modèles B événementiel contenant à la fois du non-déterminisme et des probabilités et qui permettent ainsi la description d'une large classe de systèmes. Nous étudions aussi ces modèles et nous les désignons dans tout le manuscrit par les modèles B événementiel mixtes. L'ambition de cette thèse est également de proposer des processus de développement formel qui permettent l'introduction progressive des probabilités dans un modèle B événementiel par raffinement.

La figure 1.1 donne un récapitulatif global de toutes les possibilités de développement permettant l'intégration des probabilités au sein du B événementiel. Dans cette figure, l'axe vertical représente l'introduction de plus de détails dans une machine B événementiel alors que l'axe horizontal représente l'introduction des probabilités. Plusieurs types de machines apparaîssent dans cette figure, nous les présentons dans ce qui suit :

- Les machines de l'axe vertical de gauche sont des machines B événementiel standards qui contiennent uniquement des choix non-déterministes et aucune information probabiliste ;
- Les machines de l'axe vertical de droite sont des machines B événementiel purement probabilistes où tous les choix sont probabilistes ;
- Les machines dans le centre de la figure sont des machines B événementiel mixtes qui contiennent à la fois du non-déterminisme et des probabilités.

En fonction du système à modéliser, le processus de développement peut varier d'un axe vers l'autre.

Lorsque le système n'a aucun aspect probabiliste, le processus peut rester dans l'axe vertical de gauche et se termine à la fin de cet axe. Le système est ainsi décrit par un ensemble de machines liées par raffinement.

Lorsque le système est purement probabiliste, le processus de développement peut varier. En effet, nous pouvons commencer à partir d'une machine abstraite non-déterministe, et la raffiner progressivement par ajout de nouveaux détails jusqu'à l'obtention d'une description complète du système par cette machine non-déterministe. Finalement, tous les choix non-déterministes de cette machine sont remplacés par des choix probabilistes, cette dernière étape s'appelle la probabilisation, elle est présentée par l'axe horizontal bleu en bas de la figure 1.1. Une autre possibilité pour la description d'un système purement probabiliste consiste à partir dans un premier temps d'une machine non-déterministe abstraite, et appliquer la probabilisation sur cette machine obtenant ainsi une machine abstraite purement probabiliste. Dans un second temps, il faut ajouter des détails probabilistes par raffinement probabiliste comme montré dans l'axe vertical rouge de droite dans la figure 1.1.

Lorsque le système contient à la fois des aspects probabilistes et non-déterministes, une possibilité de développement consiste à ajouter par raffinement et/ou par probabilisation partielle des informations non-déterministes et des informations probabilistes permettant ainsi l'obtention des machines B événementiel mixtes (présentée par l'axe vert).

Toutes ces possibilités de développement seront détaillées dans la suite de ce manuscrit.

Plan du Document

Le travail de recherche élaboré depuis le début de cette thèse a fait l'objet de huit chapitres répartis comme suit :

Le chapitre 2 introduit principalement la méthode B événementiel ainsi que le cadre théorique de la thèse. Dans un premier temps, nous donnons un aperçu de la méthode B événementiel ainsi que son assistant de preuve automatique Rodin, puis nous présentons l'étude de cas d'un protocole pair à pair. Dans un second temps, nous présentons les systèmes de transitions, les systèmes de transitions probabilistes, les automates probabilistes ainsi que les chaines de Markov discrètes. Nous clôturons ce chapitre par un état de l'art sur les travaux ayant déjà traité l'extension du B événementiel par des probabilités.

Le chapitre 3 présente notre première contribution qui consiste en une version purement probabiliste du B événementiel. Nous commençons par expliquer comment procéder pour introduire des probabilités au sein du B événementiel puis nous exprimons la syntaxe de notre proposition de B événementiel probabiliste. Ensuite, nous étudions la cohérence d'un modèle B événementiel probabiliste : étant donné que la cohérence est définie par des obligations de preuve, nous définissons de nouvelles obligations de preuve spécifiques au B événementiel probabiliste et proposons les adaptations nécessaires des obligations standards du B événementiel pour qu'elles puissent

être appliquées sur un modèle B événementiel probabiliste. De plus, nous étudions la sémantique d'un modèle B événementiel probabiliste exprimée en termes de chaines de Markov discrètes. Finalement, nous détaillons l'étude de cas du protocole pair à pair dans ce cadre. Notons que les machines présentées dans ce chapitre sont des machines purement probabilistes, correspondant donc à l'axe vertical de droite dans la figure 1.1.

Notre deuxième contribution est présentée au chapitre 4. Dans ce chapitre, nous étudions les modèles B événementiel contenant à la fois des probabilités et du non-déterminisme. Ce type de modèles conduit à une extension spécifique du B événementiel que nous désignons par le B événementiel mixte. Nous étudions également la cohérence des modèles B événementiel mixtes et nous exprimons leur sémantique en termes d'automates probabilistes. Nous notons que les machines présentées dans ce chapitre sont des machines mixtes correspondant donc au centre de la figure 1.1

Dans le chapitre 5, nous détaillons toutes les possibilités de développement présentées dans la figure 1.1. Nous présentons ainsi plusieurs processus de développement qui permettent l'intégration des probabilités au sein d'un modèle B événementiel. Nous commençons par présenter le processus de probabilisation qui permet de passer d'un modèle B événementiel standard vers un modèle B événementiel probabiliste. Nous étudions par la suite un cas particulier de raffinement d'un modèle B événementiel probabiliste : l'ajout de nouveaux événements probabilistes dans le cas des modèles B événementiel probabilistes. Afin de garantir la correction de ce cas particulier du raffinement, nous étudions la convergence presque certaine d'un ensemble d'événements probabilistes. Finalement, nous étendons l'ajout d'événements probabilistes au cas de modèles B mixtes ou non-déterministes(standard).

Dans le but de valider la pertinence des éléments proposés dans cette thèse, nous présentons dans le chapitre 6 deux études de cas supplémentaires. La première concerne le système de train d'atterrisage d'un avion et la deuxième concerne un système de freinage d'urgence.

Nous présentons dans le chapitre 7 notre extension à la plateforme Rodin pour prendre en compte les éléments du B événementiel probabiliste. Enfin, ce document s'achève par une conclusion générale et des perspectives.

Publications scientifiques

Les publications issues de ce travail de thèse sont les suivantes :

- *Moving from Event-B to Probabilistic Event-B*, présentée dans la conférence internationale SAC-SVT 2017
- *Une extension probabiliste pour Event-B*, présentée dans la conférence francophone AFADL 2017
- *Introducing probabilistic reasoning within Event-B*, acceptée dans le journal international SOSYM

2

B événementiel et systèmes de transitions

2.1 Introduction

De nombreux formalismes de spécification de haut niveau existent et permettent la description des comportements des systèmes. Nous présentons dans ce chapitre la méthode B événementiel et le formalisme systèmes de transitions. Le B événementiel[15, 21] est une méthode de spécification et de vérification proposée par Jean-Raymond Abrial comme une évolution de la méthode B classique[16]. Le B événementiel a gardé la puissance et la simplicité de la méthode B, mais il a apporté des améliorations sur plusieurs autres aspects. Les deux méthodes ont le même fondement mathématique[16], elles sont basées sur la logique du premier ordre et la théorie des ensembles. Le B événementiel permet la description de systèmes distribués, parallèles, multi-modals, réactifs et interactifs. Cela peut être, par exemple, un système de train[17], un protocole de communication[20] ou un système de gestion (réservation) de vol[17]. Contrairement à la méthode B, le B événementiel considère un système fermé pour représenter l'ensemble des composants dans un seul modèle. De même, le B événementiel définit le comportement dynamique du système sous forme d'événements et non par des opérations comme en B classique.

Un modèle B événementiel représente une abstraction mathématique d'un système. Il prend en compte tous les aspects du système, statique et dynamique. Les éléments statiques d'un système se placent dans un type de composant B événementiel appelé *'contexte'* alors que les aspects dynamiques d'un système se placent dans un type de composant B événementiel appelé *'machine'*.

Un contexte décrit les aspects statiques d'un système. Il peut contenir des définitions d'ensembles, des constantes, des axiomes caractérisant les constantes et des propriétés que l'on démontre.

Une machine nous montre comment le système évolue et sous quelles conditions. L'état du système est décrit dans la machine par des *variables*. L'évolution du système est capturée par les *événements*. Un événement s'exécute sous une *garde*, qui est une condition nécessaire pour que l'action de l'événement soit appliquée sur les variables du système. Ainsi, l'état du système évolue par l'exécution d'événements. L'application de l'action consiste en l'application parallèle et atomique d'un ensemble de substitutions, c'est-à-dire, des opérations modifiant les valeurs des variables du système.

Les contextes seront détaillés dans la section 2.2.1 alors que les machines seront détaillées dans la section 2.2.2. Les événements seront décrits dans la section 2.2.3 alors que les substitutions seront décrites dans la section 2.2.4.

Le B événementiel adopte l'approche de conception de systèmes *correctes par construction* [17]. Cette approche garantit que le système est correct à la fin de son développement formel. Dans cette approche, un système est initialement décrit par une spécification abstraite. Les détails sont introduits progressivement dans cette spécification jusqu'à l'obtention d'une spécification globale contenant tous les éléments du système à modéliser. Le passage d'une spécification vers une autre plus détaillée se fait par raffinement [46]. Le raffinement en B événementiel concerne les machines et les contextes. Les aspects statiques d'un système sont décrits par une suite de contextes où chaque contexte *étend* le contexte qui le précède. Les aspects dynamiques d'un système sont décrits par une suite de machines où chaque machine *raffine* la machine qui la précède. Nous notons que dans l'approche *correcte par construction*, un modèle B événementiel correspond à une spécification. Ainsi, un système en B événementiel est décrit par une suite de modèles. Chaque modèle est associé à une preuve de correction. Le passage d'un modèle à un autre est aussi associé à une preuve de correction. Nous détaillerons ces preuves dans les sections 2.3 et 2.4. Le dernier modèle dans cette suite doit offrir un niveau de détails du système à partir duquel il est possible de générer du code exécutable.

Plusieurs outils de développement permettent l'application des méthodes B et B événementiel. L'outil le plus connu est *l'AtelierB* [1] développée par la société *ClearSy* [5]. C'est le premier outil qui permet la mise en oeuvre pratique de la méthode B. *L'AtelierB* est doté de prouveurs spécifiques pour la preuve des spécifications B. Dans le même contexte, la même société a proposé l'outil *B4free* [3] pour l'enseignement de la méthode B en milieu académique. Pour le B événementiel, c'est la plateforme *Rodin* [36, 10] qui permet la spécification ainsi que la preuve des modèles B événementiel. Cette plateforme a été développée dans le cadre des projets européens *Rodin* [11] et *Deploy* [6]. Elle permet la spécification, la preuve ainsi que l'animation des modèles B événementiel. Nous notons que les outils *B4free* et *Rodin* utilisent les prouveurs de *l'AtelierB*. Un autre outil qui est souvent utilisé pour la vérification des modèles B est l'outil *ProB* [9], il permet l'animation des spécifications B et leur vérification par des techniques de *model checking*.

2.2 Modèle B événementiel

Un modèle B événementiel décrit le comportement global d'un système, il est formé par une suite de paires où chaque paire est constitué d'une machine et d'un contexte tel que le contexte est "vu" par la machine.

2.2.1 Contexte

Dans un projet de développement en B événementiel, les contextes permettent de spécifier les éléments ainsi que les propriétés statiques d'un système. Un contexte peut contenir des ensembles et des constantes. Les constantes sont des éléments dont la valeur ne change pas durant le déroulement du système étudié et qui sont définies par des axiomes. Les ensembles contiennent des éléments statiques qui ne changent pas durant le déroulement du système. Les contextes peuvent être liés entre eux. Comme une machine peut raffiner une autre machine, un contexte peut étendre un ou plusieurs contextes, et ainsi avoir accès aux éléments des précédents.

La structure d'un contexte est présentée dans la figure 2.1. La clause **EXTENDS** permet d'indiquer les contextes étendus, la clause **SETS** permet de déclarer les ensembles du contexte, la clause **CONSTANTS** permet de déclarer les constantes et la clause **AXIOMS** permet de déclarer les types des constantes. Dans le contexte proposé dans la figure 2.1, ctx correspond au nom du contexte, $\overline{\text{ctx}}$' correspond à la liste de contextes étendus par $\overline{\text{ctx}}$, $\bar{S} = \{S_1,\ldots,S_n\}$ correspond à la liste des ensembles du contexte, $\bar{c}$ correspond à la liste de constantes, $A(\bar{c})$ correspond à la liste d'axiomes et $\bar{T}(\bar{c})$ correspond à la liste des théorèmes. Les axiomes permettent, entre autres, de

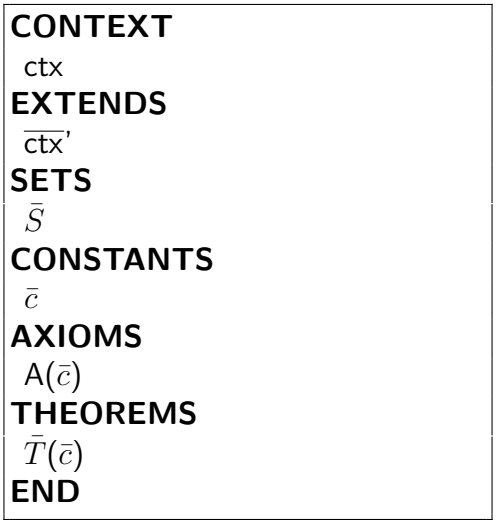

FIGURE 2.1 – Structure d'un contexte

spécifier les types de constantes. Les théorèmes permettent de spécifier des propriétés à vérifier sur les constantes du contexte. Pour une utilisation plus simple des contextes, nous dénoterons dans la suite de ce chapitre un contexte par un tuple ctx=($\bar{S}$, ctx', $\bar{c}$, A($\bar{c}$), $\bar{T}(\bar{c})$).

2.2.2 Machine

Alors que les contextes permettent la spécification des éléments statiques d'un système, une machine permet la spécification de son comportement dynamique. Ce comportement dynamique est exprimé par des événements : l'exécution d'un événement permet la modification de la valeur des variables. Les variables sont des éléments dont la valeur évolue au-fur-et-à mesure du déroulement du système. Les types des variables d'une machine B événementiel sont précisés dans l'invariant. De plus, l'invariant peut contenir des propriétés qui doivent être établies par tous les états du système. Une machine peut également contenir des théorèmes, ces théorèmes sont prouvés à partir de l'invariant. Une machine peut raffiner une autre machine, nous verrons en détail cet aspect dans la section 2.4.

La structure d'une machine est présentée dans la figure 2.2. La clause **REFINES** permet de préciser le nom de la machine raffinée, cette clause est absente lorsque la machine est abstraite et ne raffine aucune autre machine. La clause **SEES** permet de préciser les noms des contextes utilisés par cette machine. La clause **VARIABLES** permet de préciser les noms des variables de la machine. La clause **INVARIANT** permet de préciser les invariants de la machine. La clause **VARIANT** permet de préciser le variant de la machine (nous détaillerons cet élément dans la section 2.4.3). La clause **THEOREMS** permet de spécifier les théorèmes de la machine. La clause **EVENTS** permet de préciser les événements de la machine.

Dans la machine proposée dans la figure 2.2, mch correspond au nom de la machine, mch0 correspond au nom de la machine raffinée par mch (ie. mch raffine mch0), ctx correspond à la liste de contextes utilisés par mch0, $\bar{v}$ correspond à la liste des variables de la machine, I($\bar{v}$) dénote l'invariant de la machine, il précise les types des variables $\bar{v}$ ainsi que des propriétés qui doivent être satisfaites par toutes les valuations des variables $\bar{v}$. Init correspond à l'événement d'initialisation, c'est un événement particulier de la machine qui permet uniquement d'attribuer les valeurs initiales aux variables de la machine. Evts correspond à l'ensemble des événements de la machine. Pour une utilisation plus simple des machines, nous dénotons dans la suite de ce chapitre une machine par un tuple mch=($\bar{v}$, ctx, mch0, Init, I($\bar{v}$), V($\bar{v}$), Evts) et par mch=($\bar{v}$, ctx, Init, I($\bar{v}$), V($\bar{v}$), Evts) quand il s'agira d'une machine abstraite. Dans la suite, la mention du(es) contexte(s) sera souvent omise.

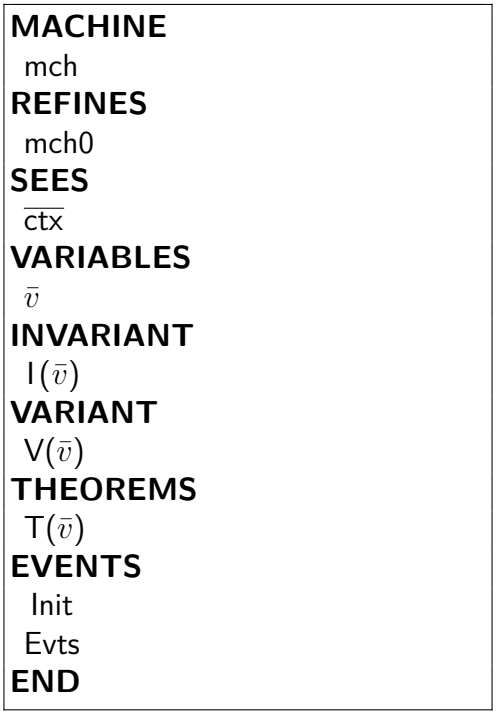

FIGURE 2.2 – Structure d'une machine

2.2.3 Événement

La dynamique d'un système est décrite en B événementiel par les événements. Ces événements agissent sur les variables du système en respectant l'invariant $I(\bar{v})$. Un événement est composé d'un nom, d'une liste de paramètres, d'une garde représentant sa condition d'activation et d'une action, i.e un ensemble de substitutions. Notons que les paramètres et la garde sont des éléments optionnels dans un événement. Selon la présence de ces éléments, nous pouvons distinguer trois types d'événements : événement *complet*, événement *gardé* ou événement *simple*. Un événement complet est un événement qui contient à la fois des paramètres, une garde et une action. Un événement gardé est un événement qui n'a pas de paramètres mais qui contient une garde et une action. Un événement simple est un événement qui contient uniquement une action. Nous présentons dans la figure 2.3 les trois formes que peut prendre un événement en B événementiel. Dans la suite, nous considérons uniquement des événements complets.

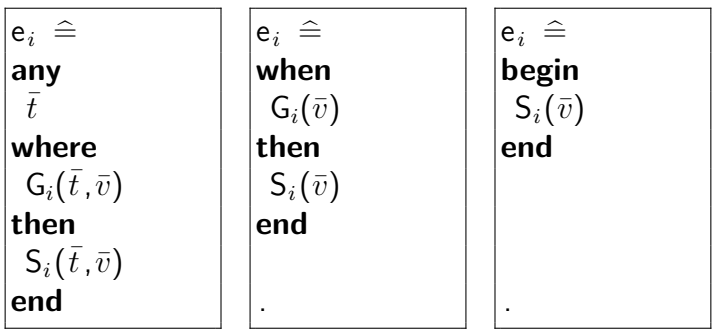

FIGURE 2.3 – Formes des événements en B événementiel

Les constituants d'un événement sont introduits par l'intermédiaire des clauses spécifiques. La clause **ANY** permet l'introduction de la liste des paramètres. La clause **WHERE** permet l'introduction de la garde. La clause **THEN** permet l'introduction de l'action. Dans les trois formes

d'événements présentées dans la figure 2.3, e_i correspond au nom de l'événement, $\bar{t}$ correspond à la liste des paramètres, $G_i(\bar{t},\bar{v})$ correspond à sa garde, i.e la condition d'activation de l'événement qui dépend des variables et des paramètres de l'événement. $S_i(\bar{t},\bar{v})$ correspond à l'action de l'événement.

2.2.4 Substitution généralisée

Nous avons mentionné que le comportement dynamique d'un système en B événementiel est décrit par l'intermédiaire des événements et qu'un événement agit sur les variables d'un système par l'intermédiaire d'une action. Une action est composée d'une liste de substitutions généralisées qui sont appliquées en parallèle formant ainsi une action atomique.

Nous distinguons trois formes de substitutions généralisées en B événementiel[17] :

- Substitution déterministe,
- Substitution non-déterministe sous forme de prédicat, et,
- Substitution énumérée non-déterministe.

Substitution déterministe Une substitution déterministe est une substitution simple qui correspond à une affectation standard des langages de programmation impératifs. Elle permet de changer la valeur d'une variable d'une machine. Étant donné x une variable et $E(\bar{t},\bar{v})$ une expression dépendante des variables de la machine et des paramètres d'un événement e_i, ce type de substitution s'écrit

$$x := E(\bar{t},\bar{v})$$

Nous notons que x et $E(\bar{t},\bar{v})$ doivent être de même type.

Ce type de substitutions peut également concerner la modification de plusieurs variables en même temps. Supposons que l'on ait des variables x_1 ... x_n auxquelles on souhaite assigner respectivement les expressions $E_1(\bar{t},v)$ ... $E_n(\bar{t},\bar{v})$. La substitution déterministe peut s'écrire :

$$x_1,x_2,\ldots,x_n := E_1(\bar{t},\bar{v}),E_2(\bar{t},\bar{v}),\ldots,E_n(\bar{t},\bar{v})$$

Substitution non-déterministe sous forme de prédicat Ce type de substitutions représente le type le plus général des substitutions non-déterministes[17]. Cette substitution modifie la valeur d'une variable x de façon à satisfaire un prédicat $Q_x(\bar{t},\bar{v},x,x')$.

Pour une variable x donnée, cette substitution est exprimée par :

$$x :| Q_x(\bar{t},\bar{v},x,x').$$

Elle signifie que la variable x prend une nouvelle valeur x' telle que le prédicat $Q_x(\bar{t},\bar{v},x,x')$ soit satisfait. Si plusieurs valeurs pour x' satisfont le prédicat $Q_x(\bar{t},\bar{v},x,x')$, alors le choix de la valeur à affecter à x est non-déterministe.

Substitution énumérée non-déterministe Dans ce type de substitutions, la variable x prend une expression parmi les expressions $E_1(\bar{t},\bar{v})\ldots E_n(\bar{t},\bar{v})$. Le choix de l'expression qui sera affecté se fait d'une façon non-déterministe. Ce choix se note au moyen de l'opérateur $:\in$ qui permet d'affecter arbitrairement une expression à partir d'un ensemble. Pour une variable x donnée pouvant prendre une valeur dans l'ensemble des expressions $\{E_1(\bar{t},\bar{v})\ldots E_n(\bar{t},\bar{v})\}$, cette substitution s'écrit

$$x :\in \{E_1(\bar{t},\bar{v})\ldots E_n(\bar{t},\bar{v})\}$$

Prédicat avant-après et action La sémantique d'une substitution en B événementiel est donnée par un prédicat avant-après (*Before-after predicate*). Ce prédicat décrit la relation entre la valeur d'une variable avant (x) et après (x') l'application de la substitution. Nous donnons dans ce qui suit ce prédicat pour chaque type de substitutions :

- Le prédicat avant-après correspondant à une substitution simple est exprimé par $x'=E(\bar{t},\bar{v})$,
- Le prédicat avant-après correspondant à une substitution non-déterministe sous forme de prédicat est exprimé par $Q_x(\bar{t},\bar{v},x,x')$,
- Le prédicat avant-après correspondant à une substitution énumérée non-déterministe est exprimé par $x' \in \{ E_1(\bar{t},\bar{v}) \ \ldots \ E_n(\bar{t},\bar{v})\}$.

Nous rappelons que l'action d'un événement en B événementiel est constituée de plusieurs substitutions qui sont exécutées en parallèle. La sémantique de l'action d'un événement est donc également décrite par un prédicat avant-après. Supposons que les variables $x_1 \ \ldots \ x_i$ soient les variables modifiées par l'action $S_i(\bar{t},\bar{v})$ et que les variables $x_{i+1} \ .. \ x_n$ soient non touchées. Supposons que $Q_{x_1}(\bar{t},x,x_1,x'_1) \ldots Q_{x_i}(\bar{t},x,x_i,x'_i)$ soient les prédicats avant-après correspondant aux substitutions modifiant les variables $x_1 \ \ldots \ x_i$. Le prédicat avant-après de l'action $S_i(\bar{t},\bar{v})$ de cet événement est exprimé par :

$$S_i(\bar{t},\bar{x},\bar{x}') \ \widehat{=} \ Q_{x_1}(\bar{t},\bar{x},x_1,x'_1) \ \wedge \ \ldots \ \wedge \ Q_{x_i}(\bar{t},\bar{x},x_i,x'_i) \wedge (x'_{i+1}=x_{i+1}) \wedge \ \ldots \ (x'_n=x_n)$$

2.3 Cohérence d'un modèle B évenementiel

La validation d'un modèle B événementiel nécessite de démontrer des obligations de preuve[17]. Les obligations de preuve sont des formules mathématiques qu'il faut démontrer pour prouver qu'un modèle B événementiel est correct.

Une obligation de preuve est construite d'un ensemble d'hypothèses et d'un but. Démontrer une obligation de preuve revient à montrer que le but peut être établi à partir des hypothèses. Plusieurs sortes d'obligations de preuve sont à considérer en B événementiel : celles liées à la preuve de correction des expressions, celles liées à la preuve de correction des machines et celles liées à la preuve de correction de raffinement. Dans un processus de spécification d'un système en B événementiel, les obligations de preuve sont générées d'une façon automatique par l'outil utilisé pour la spécification, et déchargées (en partie ou automatiquement) à l'aide des prouveurs associés à l'outil de spécification. Dans ce qui suit, nous détaillons les types d'obligations de preuve utilisés en B événementiel.

2.3.1 Correction des expressions

L'objectif de ces obligations de preuve est de s'assurer que chaque expression utilisée dans une machine ou un contexte B événementiel est correcte et bien définie. Cette expression peut être un axiome, un variant, une garde, une action, un invariant... Ces expressions peuvent être dépourvues de sens si elles ne sont pas bien définies.

Supposons que nous ayons une expression contenant l'opérateur mathématique $card$ appliqué sur un ensemble S. Nous rappelons que l'opérateur mathématique $card$ retourne le nombre d'éléments contenus dans un ensemble fini. Pour assurer que cette expression est bien définie, il faut entre autre s'assurer que l'ensemble S est fini. Nous devons donc vérifier que $finite(S)$ est satisfait où $finite$ est un opérateur mathématique qui s'applique sur un ensemble d'éléments et qui retourne vrai lorsque l'ensemble est fini et faux dans le cas contraire.

2.3.2 Obligations de preuve de machine

Concernant la dynamique d'une machine, nous dirons qu'une machine satisfait une obligation de preuve si tous ses événements satisfont cette obligation de preuve. Nous détaillons dans ce qui suit les différentes obligations de preuve.

Faisabilité des événements L'objectif de cette obligation de preuve est de s'assurer que l'action de chaque événement est applicable. Rappelons que l'action d'un événement est définie par un prédicat avant-après qui décrit la relation entre la valeur des variables avant ($\bar{v}$) et après ($\bar{v}'$) l'occurrence de l'action. L'objectif de cette obligation est de s'assurer pour chaque événement qu'il existe au moins une valuation des variables $\bar{v}'$ obtenue après son exécution telle que le prédicat avant-après est satisfait. Démontrer la faisabilité d'une action d'un événement consiste à démontrer la faisabilité de toutes les substitutions constituant cette action.

Pour un événement e_i avec une garde $\mathsf{G}_i(\bar{t},\bar{v})$ et une action $\mathsf{S}_i(\bar{t},\bar{v})$, cette obligation de preuve est présentée formellement comme suit :

$$\frac{\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v})}{\exists\, \bar{v}'.\, \mathsf{S}_i(\bar{t},\bar{v},\bar{v}')}\ \mathsf{evt/FIS}$$

Préservation de l'invariant C'est l'obligation de preuve la plus importante parmi les obligations de preuve du modèle. Nous rappelons qu'un invariant est une propriété du système qui doit être vraie dans tous les états du système. Dans une machine B, cette obligation assure que l'invariant doit être vrai après l'exécution de chaque événement de la machine : l'action de chaque événement ne viole pas l'invariant après son exécution. Pour un événement e_i avec garde $\mathsf{G}_i(\bar{t},\bar{v})$ et action $\mathsf{S}_i(\bar{t},\bar{v})$, cette obligation de preuve est présentée formellement comme suit :

$$\frac{\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \wedge \mathsf{S}_i(\bar{t},\bar{v},\bar{v}')}{\mathsf{I}(\bar{v}')}\ \mathsf{evt/INV}$$

Cas particulier de préservation de l'invariant L'initialisation est un événement particulier en B événementiel. C'est le premier événement qui s'exécute dans une machine, il permet d'initialiser toutes les variables de la machine. L'obligation de preuve de préservation de l'invariant correspondante permet de prouver que l'action de l'événement de l'initialisation établit correctement l'invariant. Formellement, cette obligation de preuve est présentée comme suit :

Init /INV
$\mathsf{S}_i(\bar{t},\bar{v},\bar{v}')$
$\vdash$
$\mathsf{I}(\bar{v}')$

Absence du blocage Cette obligation de preuve assure que dans toutes les valuations possibles des variables de la machine qui satisfont l'invariant, au moins un événement est activable. Puisque un événement n'est activable que lorsque sa garde est satisfaite, l'hypothèse de cette obligation de preuve correspond à l'invariant, le but correspond à la disjonction des gardes des événements de la machine. Cette obligation de preuve est formellement donnée comme suit.

machine/DLF
$\mathsf{I}(\bar{v})$
$\vdash$
$(\mathsf{G}_1(\bar{t},\bar{v}) \vee \ldots \vee \mathsf{G}_n(\bar{t},\bar{v}))$

Nous notons que cette obligation de preuve est optionnelle en B événementiel : en effet, elle n'a de sens que si le système spécifié ne doit pas terminer.

2.4 Raffinement

D'une façon générale, le raffinement est un processus qui permet l'enrichissement ou la modification d'une spécification par l'ajout de nouvelles fonctionnalités ou par l'ajout de nouveaux détails aux fonctionnalités décrites dans la spécification initiale. Comme nous l'avons précisé dans la section 2.1, le processus de développement en B événementiel est basé sur le raffinement.

Le concept de raffinement a été introduit initialement dans les années 1970 par Dijkstra . Il a été ensuite formalisé par Back[23, 24] dans les années 1980. Depuis ce moment, plusieurs travaux de recherche ont étudié ce concept. Nous citons à titre d'exemples, les travaux de Abadi et Lamport[13], Back[26], Morgan[77], Morris[80] et Abrial[16].

2.4.1 Raffinement en B événementiel

Le raffinement en B événementiel consiste en l'extension des contextes et le raffinement des machines. L'extension du contexte consiste en l'ajout de nouveaux éléments statiques : ensembles, constantes, axiomes... Le raffinement de machine permet de décrire d'une manière progressive le comportement dynamique du système. Il consiste à transformer une machine abstraite en une machine plus concrète par l'ajout de nouveaux éléments, ce qui rapproche d'une implantation par un langage de programmation.

Le raffinement offre plusieurs avantages. D'abord, il permet le partage de la complexité de développement formel : au lieu de spécifier un modèle global contenant tous les artefacts du système, le raffinement permet l'introduction progressive du comportement du système. Cela permet la prise en compte de tous les aspects du système et facilite la compréhension du modèle. D'un autre côté, il permet aussi la simplification de la preuve de correction des machines B événementiel et la réutilisation des propriétés déjà prouvées.

En B événementiel, une machine concrète N=($\bar{w}$,J($\bar{v}$,$\bar{w}$),Evts$_c$,Init$_c$) peut raffiner une et une seule machine abstraite M=($\bar{v}$,I($\bar{v}$),Evts$_a$, Init$_a$), le raffinement est décrit dans la machine concrète N. Dans N, $\bar{w}$ correspond à l'ensemble des variables concrètes. Nous notons que les variables de la machine abstraite peuvent être conservées dans la machine concrète ou disparaître lors du raffinement. J($\bar{v}$,$\bar{w}$) est l'invariant de collage, il permet la liaison entre les variables abstraites et les variables concrètes. Evts$_c$ est l'ensemble des événements concrets, Init$_c$ est l'événement d'initialisation concret. La correction de passage d'une machine abstraite vers une machine concrète par raffinement est vérifié par un ensemble d'obligations de preuves, nous les détaillerons dans la section 2.4.3.

Le raffinement en B événementiel consiste en :

- L'ajout de nouvelles variables et/ou le remplacement des variables abstraites,
- Le raffinement des événements asbtrait,
- L'ajout de nouveaux événements.

L'ajout de nouvelles variables ou le remplacement des variables abstraites induit l'ajout de nouveaux invariants et/ou la modification des invariants abstraits. Pour le raffinement des événements abstrait, chaque événement d'une machine abstraite doit être raffiné par un ou plusieurs événements concrets (*split*). Réciproquement, un ou plusieurs événements abstraits de la machine abstraite peuvent être raffinés par un unique événement concret (*merge*).

Note 1 *Dans la suite de notre travail, nous ne traitons pas les cas de* split *et de* merge*, nous traitons uniquement le cas de raffinement direct d'un événement abstrait par un événement concret.*

Pour ce type de raffinement, un événement abstrait peut être raffiné par un événement concret par l'ajout, la modification voire la suppression de paramètres, gardes ou actions. Nous donnons dans la figure 2.4 la structure d'un événement abstrait e$_a$ et d'un événement concret e$_c$.

e$_a$ $\widehat{=}$ **any** $\bar{t}$ **where** G$_a$($\bar{t}$,$\bar{v}$) **then** S$_a$($\bar{t}$,$\bar{v}$) **end** .	e$_c$ $\widehat{=}$ **refines** e$_a$ **any** $\bar{u}$ **where** G$_c$($\bar{u}$,$\bar{w}$) **with** W$_c$($\bar{t}$,$\bar{u}$,$\bar{v}$,$\bar{v}$',$\bar{w}$,$\bar{w}$') **then** S$_c$($\bar{u}$,$\bar{w}$) **end**

FIGURE 2.4 – Structure d'un événement abstrait et d'un événement concret

L'événement e$_c$ raffine l'événement e$_a$. Dans la structure de l'événement e$_c$, nous remarquons l'apparition d'un nouvel élément W$_c$($\bar{t}$,$\bar{u}$,$\bar{v}$,$\bar{v}$',$\bar{w}$,$\bar{w}$'). Cet élément s'appelle le témoin (witness), il est ajouté dans le cas de disparition de paramètres abstraits lors du raffinement. Il permet d'établir un lien entre les paramètres abstraits disparus et le corps de l'événement concret.

L'ajout de nouveaux événements représente un aspect important du raffinement des machines en B événementiel. L'introduction de nouveaux événements dans une machine permet l'introduction de nouveaux comportements. Ces nouveaux événements doivent vérifier les obligations de

preuve de cohérence présentées dans la section 2.3. De plus, il faut s'assurer que leur introduction ne contredit pas le comportement de la machine abstraite. Pour cela, il faut prouver que les nouveaux événements ajoutés ne divergent pas, c'est-à-dire qu'ils ne prennent pas le contrôle de la machine concrète indéfiniment. Il faut donc prouver que les nouveaux événements introduit durant le raffinement convergent. Il faut également prouver que les nouveaux événements introduit par raffinement n'introduisent pas de nouveaux blocages.

2.4.2 Exactitude du raffinement

Une machine abstraite est liée à une machine concrète par une relation de raffinement. En B événementiel, la validation du passage d'une machine abstraite vers une machine concrète est vérifiée par un ensemble d'obligations de preuve. Nous présentons dans ce qui suit les obligations de preuve dédiées à la validation de raffinement. Supposons que nous avons une machine B $\mathsf{M}=(\bar{v},\mathsf{I}(\bar{v}),\mathsf{Evts}_a,\mathsf{Init}_a)$ raffinée par une machine concrète $\mathsf{N}=(\bar{w},\mathsf{J}(\bar{v},\bar{w}),\mathsf{Evts}_c,\mathsf{Init}_c)$. Nous notons que les deux machines M et N vérifient bien les obligations de preuve de cohérence présentées dans la section 2.3. Nous présentons dans ce qui suit les obligations de preuve spécifiques à la correction du raffinement.

Correction des témoins (witness) Lors du raffinement d'un événement abstrait non-déterministe par un événement concret, il est possible que certains paramètres de l'événement abstrait disparaissent. Dans ce cas, il faut définir un témoin (witness) pour prouver la correction de l'événement. Comme présenté dans la figure 2.4, un témoin $\mathsf{W}_c(\bar{t},\bar{u},\bar{v},\bar{v}',\bar{w},\bar{w}')$ lie les paramètres de l'événement abstrait aux paramètres et aux variables de l'événement concret. Pour l'événement e_c, cette obligation de preuve est présentée formellement comme suit.

$$\frac{\mathsf{I}(\bar{v}) \wedge \mathsf{J}(\bar{v},\bar{w}) \wedge \mathsf{G}_c(\bar{u},\bar{w})}{\exists\, \bar{t}.\ \mathsf{W}(\bar{t},\bar{u},\bar{v},\bar{v}'\bar{w},\bar{w}')}\ \mathsf{evt/WFIS}$$

Renforcement de la garde L'objectif de cette obligation de preuve est de s'assurer que la garde de l'événement concret est au moins aussi forte que la garde de l'événement abstrait correspondant. Informellement, cette obligation de preuve garantie que dans tous les états où nous pouvons activer un événement concret, nous pouvons activer l'événement abstrait correspondant.

Pour les événements e_a et e_c présentés dans la figure 2.4, cette obligation de preuve est présentée formellement comme suit.

grd/STRENGTH
$\mathsf{I}(\bar{v}) \wedge$
$\mathsf{J}(\bar{v},\bar{w}) \wedge$
$\mathsf{G}_c(\bar{u},\bar{w}) \wedge$
$\mathsf{W}_c(\bar{t},\bar{u},\bar{v},\bar{v}',\bar{w},\bar{w}')$
$\vdash$
$\mathsf{G}_a(\bar{t},\bar{v})$

Simulation des actions L'objectif de cette obligation de preuve est de s'assurer que l'action de chaque événement concret simule l'action de l'événement abstrait correspondant. Il assure qu'au moment de l'exécution d'un événement concret, l'action concrète ne contredit pas l'action de l'événement abstrait correspondant. Pour les événements e_a et e_c présentés dans la figure 2.4, cette obligation de preuve est présentée formellement comme suit.

act/SIM
$\mathsf{I}(\bar{v}) \wedge$
$\mathsf{J}(\bar{v},\bar{w}) \wedge$
$\mathsf{G}_c(\bar{u},\bar{w}) \wedge$
$\mathsf{S}_c(\bar{u},\bar{w},\bar{w}') \wedge$
$\mathsf{W}_c(\bar{t},\bar{u},\bar{v},\bar{v}',\bar{w},\bar{w}')$
$\vdash$
$\mathsf{S}_a(\bar{t},\bar{v},\bar{v}')$

2.4.3 Convergence des nouveaux événements

Comme mentionné précédemment, une opération importante du raffinement des machines en B événementiel consiste en l'ajout de nouveaux événements. Cette opération permet la description de nouveaux comportements. Cependant, afin de préserver certaines propriétés de la machine abstraite dans laquelle nous introduisons les événements, il faut s'assurer que leur introduction ne contredit pas le comportement de la machine abstraite. Pour cela, il faut prouver que les nouveaux événements ajoutés ne prennent pas le contrôle de la machine concrète indéfiniment. Pour cela, une solution est de proposer un variant, et de prouver que l'action de chaque nouvel événement décroît la valeur de ce variant. Ainsi, on montrera bien que les nouveaux événements finiront par terminer. Cela est formalisée par deux obligations de preuve : la première veille à ce que le variant proposé soit bien un entier positif minoré par zéro. La deuxième obligation assure que chaque nouvel événement décroît le variant. Nous présentons dans ce qui suit ces deux obligations de preuve.

Variant minoré L'objectif de cette obligation de preuve est de s'assurer que sous l'invariant et la garde de chaque nouvel événement, le variant est bien un entier naturel strictement positif, c'est-à-dire minoré par 0. Pour un nouvel événement e_c, cette obligation de preuve est présentée formellement comme suit.

$$\frac{\mathsf{I}(\bar{v}) \wedge \mathsf{G}_c(\bar{t},\bar{v})}{\mathsf{V}(\bar{v}) \in \mathsf{NAT1}} \quad \text{evt/var/NAT}$$

Convergence L'objectif de cette obligation de preuve est de s'assurer que l'action de chaque nouvel événement convergent doit toujours décroître le variant. Pour chaque nouvel événement e_c, cette obligation de preuve est présentée formellement comme suit.

$$\frac{\mathsf{I}(\bar{v}) \wedge \mathsf{G}_c(\bar{t},\bar{v})}{\forall\, \bar{v}'.\, \mathsf{S}_c(\bar{t},\bar{v},\bar{v}') \Rightarrow \mathsf{V}(\bar{v}') < \mathsf{V}(\bar{v})} \quad \text{evt/Var}$$

2.5 Étude de cas : Protocole Pair à Pair

Pour illustrer la méthode B événementiel, nous spécifions un protocole pair à pair inspiré de celui de Bittorent [4]. Une description complète de ce protocole est présentée dans [8]. Ce cas d'étude comporte N clients qui essayent de télécharger un fichier partitionné en K blocs. Nous notons que N et K sont strictement supérieurs à zéro. Initialement, aucun client n'a encore téléchargé aucun bloc. Le protocole se termine lorsque tous les clients ont réussi à télécharger tous les blocs du fichier. La figure 2.5 illustre le principe général du protocole.

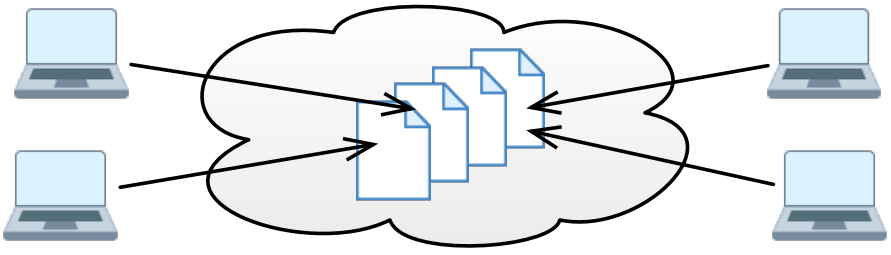

FIGURE 2.5 – Principe général du protocole

2.5.1 Stratégie du développement du protocole en B événementiel

Nous rappelons que le processus de développement en B événementiel est basé sur le raffinement. Ainsi, nous utilisons le raffinement pour une description progressive du protocole. Dans cette partie, nous expliquons notre stratégie de raffinement. Dans la figure 2.6, nous présentons la stratégie du développement du protocole en B événementiel. Nous créons un premier contexte protocole1 et

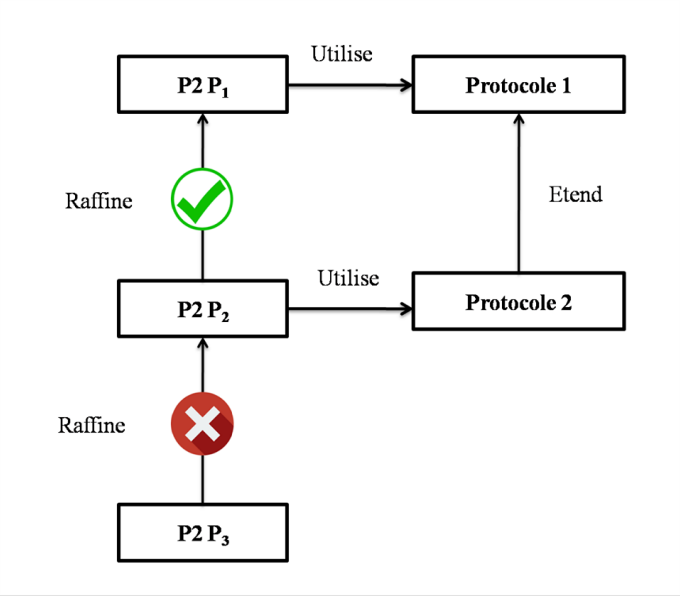

FIGURE 2.6 – Stratégie du développement du protocole en B événementiel

une première machine $P2P_1$ qui utilise les éléments de ce contexte. Dans cette machine, nous spécifions l'objectif global du protocole sans spécifier le détail. Nous étendons le contexte protocole1 obtenant ainsi le contexte protocole2 et nous raffinons la première machine $P2P_1$ obtenant ainsi la machine $P2P_2$. Dans cette machine, nous détaillons le téléchargement d'un bloc par un client. Nous raffinons la machine $P2P_2$ obtenant ainsi la machine $P2P_3$. Dans cette machine, nous spécifions les cas d'erreurs qui peuvent arriver durant le téléchargement d'un bloc par un client. Nous allons montrer que l'étape de raffinement de $P2P_2$ par $P2P_3$ n'est pas correcte.

2.5.2 La première machine

Le protocole décrit le téléchargement d'un fichier partitionné en K blocs par un ensemble de N clients. Ainsi, nous avons besoin de deux constantes, N pour dénoter le nombre de clients et K pour dénoter le nombre de blocs. De même, nous avons besoin de constantes pour décrire l'état d'un bloc par rapport à un client, c'est-à-dire si ce client a réussi à télécharger ce bloc ou si ce n'est pas encore le cas. Tous les éléments de ce contexte sont stockés dans le contexte protocole présenté dans la figure 2.7.

La première machine présentée dans la figure 2.7 permet de spécifier l'objectif général du protocole d'une manière très abstraite. Dans cette machine, l'état du système est décrit par une seule variable DB. DB est une matrice qui indique pour chaque client $c \in 1..N$ et chaque bloc $k \in 1..K$ si le client c a réussi à télécharger ce bloc ($DB(c \mapsto k)$=finished) ou si ce n'est pas encore le cas ($DB(c \mapsto k)$=empty). Initialement, aucun client n'a encore téléchargé aucun bloc. Cette machine contient un seul événement : AllDL. Cet événement modélise le téléchargement de tous les blocs par tous les clients, il agit sur la variable DB, cette dernière prend une nouvelle valeur telle que pour tous les clients $c \in 1..N$ et pour tous les blocs $k \in 1..K$ nous avons $DB(c \mapsto k)$=finished.

2.5.3 Deuxième machine

La deuxième machine raffine la première décrite en figure 2.7. Cette machine utilise le contexte protocole1 donnée dans la figure 2.8, ce dernier est une extension du contexte protocole donnée dans la figure 2.8. Dans ce contexte protocole1, nous redéfinissons l'ensemble $DB-Status$ pour ajouter un nouvel état intermédiaire incoming indiquant que le client est en train de télécharger le

CONTEXT
protocole
CONSTANTS
N
K
empty
finished
SETS
DB−STATUS
AXIOMS
N∈ NAT1
K∈ NAT1
partition (DB−STATUS,{empty},{finished})
END

MACHINE
$P2P_1$
SEES
protocole
VARIABLES
DB
INVARIANTS
DB∈ 1..N × 1..K → {empty,finished}
EVENTS
Init ≙
begin
DB:=(1..N × 1..K) × {empty}
end

AllDL ≙
any magicDB **where**
DB = (1..N × 1..K) × {empty} ∧
magicDB ∈ 1..N × 1..K → {empty,finished} ∧
magicDB = (1..N × 1..K) × {finished}
then
DB := magicDB
end

END

FIGURE 2.7 – Première machine

CONTEXT
protocole1
EXTENDS
protocole
CONSTANTS
N
K
empty
finished
incoming
SETS
DB−STATUS
AXIOMS
N∈ NAT1
K∈ NAT1
partition (DB−STATUS,{empty},{incoming},{finished})
END

FIGURE 2.8 – Deuxième contexte

bloc mais n'a pas encore fini.

Dans la machine $P2P_2$ donnée dans la figure 2.9, nous ajoutons une nouvelle variable DBin qui représente l'état du téléchargement. Elle permet de représenter l'état du téléchargement d'un bloc par un client au cours de l'exécution de la machine. Pour chaque client c∈ 1..N et chaque bloc b ∈1..K, cet état peut être finished signifiant que le client c a réussi à télécharger le bloc b, incoming signifiant que le client c est en train de télécharger le bloc b ou empty signifiant que le client c n'a

MACHINE
P2P$_2$
SEES
protocole1
REFINES
P2P$_1$
VARIABLES
DB
DBin
INVARIANTS
DBin $\in$ 1..N $\times$ 1..K $\rightarrow$ {empty,incoming,finished} $\wedge$
$\forall$ c . (c $\in$ 1..N $\Rightarrow$
card({b | b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming}) $\leq$ 1)
VARIANT
2 $\times$ N $\times$ K
$-$ 2 $\times$ card({c$\mapsto$b | c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = finished})
$-$ card({c$\mapsto$b |c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming})
EVENTS
Init $\widehat{=}$
begin
DB := (1..N $\times$ 1..K) $\times$ {empty} ||
DBin := (1..N $\times$ 1..K) $\times$ {empty}
end

DLFinished $\widehat{=}$
refines AllDL
when
DB = (1..N $\times$ 1..K) $\times$ {empty} $\wedge$
DBin = (1..N $\times$ 1..K) $\times$ {finished}
then
DB := DBin
end

Start1DL $\widehat{=}$
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$
DBin(c$\mapsto$b) = empty $\wedge$
card({k | k $\in$ 1..K $\wedge$ DBin(c$\mapsto$k) = incoming})=0
then
DBin(c$\mapsto$b) := incoming
end

Finish1DL $\widehat{=}$
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$
DBin(c$\mapsto$b) = incoming
then
DBin(c$\mapsto$b) := finished
end
END

FIGURE 2.9 – Deuxième machine

pas encore commencé le téléchargement du bloc b.

Nous introduisons également deux nouveaux événements, Start1DL et Finish1DL.

- Dans Start1DL, un client c et un bloc b sont choisis d'une manière non-déterministe de telle sorte que le client c n'a pas encore commencé le téléchargement du bloc b. Son action déclenche le début de téléchargement de b par c (DBin(c↦b):=incoming).
- L'événement Finish1DL décrit la terminaison du téléchargement d'un bloc b par un client c. Pour cela, il faut que le client c ait commencé le téléchargement du bloc b.

Finalement, l'événement DLFinished est un raffinement de l'événement AIIDL de la première machine. Cet événement est activé lorsque DBin= (1..N × 1..K) × finished, c'est-à-dire lorsque tous les clients ont réussi à télécharger tous les blocs. Il affecte la valeur de DBin à DB pour réaliser la substitution abstraite de l'événement AIIDL.

Comme nous introduisons deux nouveaux événements Start1DL et Finish1DL, nous devons montrer leur convergence. Ainsi, nous avons introduit un nouveau variant :

$$2 \times N \times K - 2 \times \mathsf{card}(\{c{\mapsto}b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c{\mapsto}b) = \mathsf{finished}\}) \\ - \mathsf{card}(\{c{\mapsto}b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c{\mapsto}b) = \mathsf{incoming}\})$$

Ce variant doit être décru par les actions de Start1DL et Finish1DL. Pour l'événement Start1DL, son action déclenche le téléchargement d'un bloc b par un client c, et ainsi augmente le nombre d'éléments de l'ensemble $\{c{\mapsto}b \mid n{\in}1..N \wedge b \in 1..K \wedge DBin(c{\mapsto}b) = \mathsf{incoming}\}$, ce qui diminue la valeur du variant. Pour l'événement Finish1DL, son action termine le téléchargement d'un bloc b par un client c, et ainsi, augmente le nombre d'éléments de l'ensemble :

$$\{c{\mapsto}b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c{\mapsto}b) = \mathsf{finished}\},$$

ce qui diminue la valeur du variant.

Puisque l'événement Start1DL représente le déclenchement du téléchargement d'un bloc par un client et l'événement Finish1DL représente la finalisation de téléchargement d'un bloc par un client, nous concluons que la démonstration de la convergence des nouveaux événements Start1DL et Finish1DL permet de montrer la terminaison du protocole.

2.5.4 Troisième machine

À ce niveau de raffinement, nous prenons en compte certaines erreurs qui peuvent arriver durant le téléchargement d'un bloc par un client :

- l'interruption du téléchargement ;
- l'impossibilité du téléchargement du bloc à un instant donné.

Afin de formaliser ces deux erreurs, nous introduisons par raffinement un nouvel événement FailureDL. Cet événement est présenté dans la figure 2.10. Cet événement choisit un client c et un bloc b tel que le client c a commencé le téléchargement du bloc b (DBin(c ↦ b)=incoming). L'action de cet événement choisit d'une façon non-déterministe empty ou incoming pour être affecté à DBin(c↦b). empty est choisi dans le cas où l'erreur interrompt le processus de téléchargement du bloc alors que incoming est choisi dans le cas où l'erreur bloque le téléchargement.

Cet événement a la même garde que l'événement Finish1DL. Ils peuvent etre activés en même temps, le choix entre ces deux événements est non-déterministe. Puisque FailureDL est un nouvel événement introduit par raffinement, nous devons prouver sa convergence. Autrement, nous devons prouver que son action décroît la valeur du variant. Nous rappelons que le variant de cette machine $P2P_3$ est donné par :

$$2 \times \mathsf{N} \times \mathsf{K} - 2 \times \mathsf{card}(\{\mathsf{c} \mapsto \mathsf{b} \mid \mathsf{c} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\mathsf{c} \mapsto \mathsf{b}) = \mathsf{finished}\}) - \mathsf{card}(\{\mathsf{c} \mapsto \mathsf{b} \mid \mathsf{c} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\mathsf{c} \mapsto \mathsf{b}) = \mathsf{incoming}\})$$

Pour un client c et un bloc b telle que c est train de télécharger b ($\mathsf{DBin}(\mathsf{c} \mapsto \mathsf{b}) = \mathsf{incoming}$), l'action de cet événement choisit d'une façon non-déterministe empty ou incoming pour être affecté à $\mathsf{DBin}(\mathsf{c} \mapsto \mathsf{b})$. Si empty est choisi, le nombre d'éléments de l'ensemble $\{\mathsf{c} \mapsto \mathsf{b} \mid \mathsf{n} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\mathsf{c} \mapsto \mathsf{b}) = \mathsf{incoming}\}$ va diminuer, ce qui augmente la valeur du variant. Si incoming est choisi, le nombre d'éléments de l'ensemble $\{\mathsf{c} \mapsto \mathsf{b} \mid \mathsf{c} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\mathsf{c} \mapsto \mathsf{b}) = \mathsf{incoming}\}$ reste intacte, et ainsi la valeur du variant n'est pas modifiée. Nous déduisons ainsi que l'événement FailureDL ne converge pas, et ainsi, le protocole ne termine pas dans cette version. Nous montrons dans la suite de ce manuscrit comment nous procédons pour faciliter la preuve de convergence de nouveaux événements introduits par raffinement.

2.6 Systèmes de transitions

Les systèmes de transitions constituent un formalisme classique permettant de modéliser le comportement d'un système. Ce formalisme se base sur la notion d'états et de transitions. Les états sont typiquement les valeurs des variables du système et les transitions sont les actions permettant de modifier ces valeurs. Ce formalisme étant bien étudié, il est commun d'exprimer la sémantique d'autres modèles dans le cadre des sytèmes de transitions, c'est le cas par exemple pour le B événementiel. Nous introduisons donc ci-après les définitions nécessaires dans ce cadre pour exprimer la sémantique opérationnelle de nos modèles en termes de systèmes de transitions (probabilistes).

2.6.1 Systèmes de transitions

Nous commençons par donner la définition d'un système de transitions.

Définition 1 *(Système de transitions [28])*
Un système de transitions est un tuple $M = (S, S_0, Acts, T)$ avec :

- *S un ensemble d'états,*
- *$S_0 \subseteq S$ un ensemble d'états initiaux,*
- *$Acts$ un ensemble d'actions permettant d'étiqueter les transitions,*
- *$T \subseteq S \times Acts \times S$ est une relation de transition.*

Étant donnés deux états s, t et une action α, une transition $(s, \alpha, t) \in T$ qui permet de mener le système de l'état s vers l'état t et qui est effectuée par l'intermediaire de l'action α est notée $s \xrightarrow{\alpha} t$. Un système de transitions peut être fini ou infini. On dira qu'il est fini s'il est à nombre d'états et de transitions finis, c'est-à-dire si les ensembles S et $Acts$ sont finis. Autrement, on dira que le système est infini.

Un sytème de transitions peut être déterministe ou non-déterministe. Il est déterministe si la relation de transition est une fonction et non-déterministe dans le cas contraire, c'est-à-dire, pour un état donné, plusieurs transitions sortantes sont présentes. Le choix de la transition à exécuter au moment de l'exécution se fait d'une manière non-déterministe.

MACHINE
$P2P_3$
SEES
protocole1
REFINES
$P2P_1$
VARIABLES
DB
DBin
INVARIANTS
$DBin \in 1..N \times 1..K \rightarrow \{empty,incoming,finished\} \wedge$
$\forall c \cdot (c \in 1..N \Rightarrow$
$card(\{b \mid b \in 1..K \wedge DBin(c \mapsto b) = incoming\}) \leq 1)$
VARIANT
$2 \times N*K$
$- 2 \times card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = finished\})$
$- card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming\})$
EVENTS
Init $\widehat{=}$
begin
$DB := (1..N \times 1..K) \times \{empty\} \parallel$
$DBin := (1..N \times 1..K) \times \{empty\}$
end
DLFinished $\widehat{=}$
refines AllDL
when
$DB = (1..N \times 1..K) \times \{empty\} \wedge$
$DBin = (1..N \times 1..K) \times \{finished\}$
then
$DB := DBin$
end
Start1DL $\widehat{=}$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge$
$DBin(c \mapsto b) = empty \wedge card(\{k \mid k \in 1..K \wedge DBin(c \mapsto k) = incoming\}) = 0$
then
$DBin(c \mapsto b) := incoming$
end
Finish1DL $\widehat{=}$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming$
then
$DBin(c \mapsto b) := finished$
end
FailureDL $\widehat{=}$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming$
then
$DBin(c \mapsto b) :\in \{empty,incoming\}$
end
END

FIGURE 2.10 – Troisième machine

Exemple d'un système de transitions La figure 2.11 donne une représentation graphique d'un système de transitions décrivant le comportement global du protocole pair à pair. Une variable s indique l'état de téléchargement (terminé ($\checkmark$) ou non ($\sim$)). Ce système contient deux états qui donnent la valeur de cette variable s. Partant de l'état ($s = \sim$), la transition étiquetée par AllDL mène le système vers l'état ($s = \checkmark$), représentant ainsi la fin du téléchargement. La transition étiquetée par FailureDl boucle sur l'état ($s = \sim$) représentant ainsi le cas de l'erreur durant le téléchargement.

FIGURE 2.11 – Exemple d'un système de transitions

Nous présentons dans ce quit suit l'intégration des probabilités dans les systèmes de transitions.

2.6.2 Systèmes de transitions et probabilités

Comme mentionné précédemment, certains systèmes de transitions permettent la présence du non-déterminisme.

Dans certains types de systèmes de transitions, les transitions sont pondérées par des probabilités. Ce type de système ne contient alors pas de non-déterminisme, puisque la distribution de probabilités est entièrement déterminée par l'état actuel du système. Ce type de systèmes de transitions correspond au modèle de Larsen et Skou pour modéliser les systèmes probabilistes ayant un espace d'états fini. Ce modèle est appelé initialement "système probabiliste réactif" [95], puis "*fully probabilistic*" [27]. Récemment, il est nommé "chaîne de Markov à temps discret" [70].

D'autres modèles ont été proposés. Contrairement aux modèles initialement proposés par Larsen et Skou, ces modèles admettent plusieurs distributions de probabilités pour un même état de départ. À partir de cet état, un choix non-déterministe est effectué pour choisir une distribution de probabilité, une fois choisie, il y aura un choix probabiliste entre les états issus de cette distribution. Dans la littérature, ces modèles sont appelés initialement "génératifs" [95], puis "probabilistes concurrents" [27] et récémment processus de décision Markovien ou Automates probabilistes [70].

Nous présentons dans ce qui suit les définitions de ces modèles en commençant par la notion la plus générale de systèmes de transitions probabilistes. Pour un ensemble donné S, nous dénotons par $Dist(S)$ l'ensemble des distributions de probabilité sur l'ensemble S, i.e. l'ensemble des fonctions $\delta : S \to [0, 1]$ telles que $\sum_{s' \in S} \delta(s') = 1$. Nous dénotons aussi par $Card(S)$ le cardinal de l'ensemble S, c'est-à-dire, le nombre d'éléments qui apparaissent dans cet ensemble.

Nous présentons maintenant la définition d'un système de transitions probabiliste.

Définition 2 *(Système de transitions probabiliste [28])*
Un système de transitions probabiliste est un tuple $M = (S, s_0, Acts, T)$ *avec :*

- S *un ensemble d'états,*
- $s_0 \in S$ *l'état initial,*
- $Acts$ *l'ensemble des noms d'actions, et*
- $T \subseteq S \times Acts \times (S \to [0, 1])$ *une relation de transition probabiliste.*

Notons que dans le cas d'un système de transitions probabiliste, la somme des probabilités $\delta(s')$ associées à une transition (s, a, δ) dans chaque état ne fait pas nécessairement 1.

En restreignant la relation de transition T, on obtient les définitions d'un automate probabiliste et d'une chaîne de Markov discrète. Nous les présentons dans ce qui suit.

Définition 3 *(Automate probabiliste [85])*

Un automate probabiliste $M = (S, s_0, Acts, T)$ est un système de transitions probabiliste où la relation de transitions T est telle que :

$$T \subseteq S \times Acts \times Dist(S), \textit{ i.e. } \forall\, (s, a, \delta) \in T,\ \textstyle\sum_{s' \in S} \delta(s') = 1$$

Un automate probabiliste est donc un système de transitions probabiliste où la somme des probabilités associées à chaque transition vaut 1.

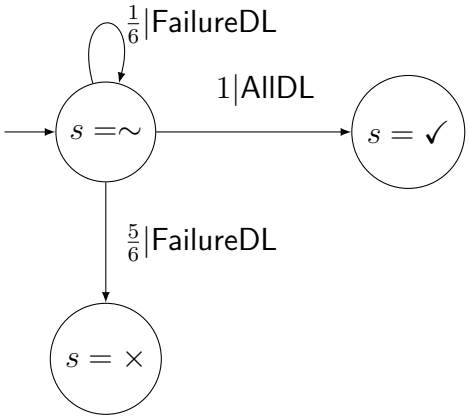

FIGURE 2.12 – Exemple d'un automate probabiliste

Exemple d'un automate probabiliste Dans la figure 2.12, nous présentons un exemple d'un automate probabiliste correspondant au protocole pair à pair. Nous remarquons l'apparition d'un nouvel état ($s = \times$), cet état représente l'interruption du téléchargement. Nous remarquons aussi l'apparition d'une nouvelle transition de l'état ($s = \sim$) vers l'état ($s = \times$) qui représente un téléchargement interrompu. Cette transition correspond à l'exécution de l'action FailureDl et elle s'exécute avec une valeur de probabilité de $\frac{5}{6}$. Nous remarquons que la somme des probabilités des transitions correspondantes à l'action FailureDL est de 1 et que la probabilité de la transition annoté par AllDL est de 1. Nous sommes donc bien dans le cadre d'un automate probabiliste.

Nous donnons maintenant la définition d'une chaîne de Markov à temps-discret.

Définition 4 *(Chaîne de Markov à temps-discret [28]) Une Chaîne de Markov à temps discret $M = (S, s_0, Acts, T)$ est un automate probabiliste avec une contrainte supplémentaire sur la relation T. Cette contrainte est donnée par :*

$$\forall\, s \in S,\ card(\{a \in Acts | \exists \delta \in Dist(S), (s, a, \delta) \in T\}) = 1$$

Une chaîne de Markov à temps discret est donc un automate probabiliste sans choix non-déterministe. Dans notre travail, nous avons besoin de faire apparaitre les actions sur les transitions d'une chaîne de Markov. Nous préférons alors utiliser une définition différente de celle donnée précédemment mais qui reste équivalente.

Définition 5 *Une chaîne de Markov à temps discret $M = (S, s_0, Acts, P)$ est un automate probabiliste où la relation de transition probabiliste est remplacée par la fonction de transition P :*

$$P : S \to Dist(Acts \times S) \textit{ telle que } \forall\, s,\ P(s) = \delta \textit{ avec } \textstyle\sum_{e \in Acts, s' \in S} \delta(e, s') = 1.$$

Exemple d'une chaîne de Markov discrète Dans la figure 2.13, nous présentons un exemple de chaîne de Markov discrète correspondant au protocole pair à pair. Cette chaîne possède les mêmes états ainsi que les mêmes transitions que le système de transitions présenté dans la figure 2.11. La différence concerne les transitions qui sont ici pondérées par des valeurs de probabilité. La probabilité d'aller de ($s = \sim$) vers ($s = \checkmark$) est de $\frac{1}{2}$ (probabilité de réussite de téléchargement) alors que la probabilité de boucler sur l'état ($s = \sim$) est de $\frac{1}{2}$ (probabilité d'erreur). Nous remarquons que la somme des probabilités de transitions sortantes de ($s = \sim$) fait bien 1, ce qui définit une chaine de Markov discrète.

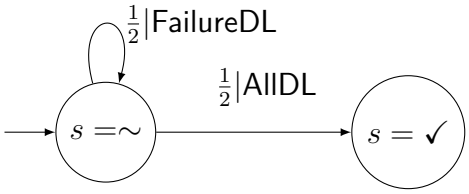

FIGURE 2.13 – Exemple d'une chaîne de Markov discrète

2.6.3 Sémantique opérationnelle d'une machine B événementiel

La sémantique opérationnelle d'une machine M=($\bar{v}$,I($\bar{v}$),Evts, Init) est donnée en termes de système de transitions $M = (S, s_0, Acts, T)$ (adaptée de [32]). L'obtention de la sémantique est comme suit :

- Les états S correspondent aux valuations des variables de la machine. Considérons le cas où nous avons une machine M contenant n variables v_i chacune ayant un domaine D_i. S est alors formé par les valuations du n-uplet (v_1,v_2,...,v_n) qui sont dans $D_1 \times D_2 \times ... \times D_n$;
- L'état s_0 correspond aux valuations des variables obtenues après l'exécution de l'événement d'initialisation Init ;
- Acts correspond à l'ensemble des noms des événements Evts de la machine M ;
- $T \subseteq S \times Acts \times S$ est l'ensemble de transitions. Il correspond à l'ensemble des exécutions possibles des différentes événements de la machine M.

Afin de bien expliquer la construction de la sémantique d'une machine B événementiel, nous présentons dans ce qui suit la sémantique opérationnelle de la machine $P2P_2$ présentée dans la section 2.5.

Sémantique opérationnelle de la machine $P2P_2$ La figure 2.14 donne un extrait du système de transitions correspondant à la machine $P2P_2$ pour N=2 et K=2. Dans cet extrait, les états représentent le contenu des variables de la machine. Pour les variables DBin et DB, les clients sont représentées en ligne et les blocs sont représentés en colonne. Le contenu des matrices est représenté par les symboles (.), (○), (●) où (.) signifie que le client n'a pas encore commencé le téléchargement du bloc, (○) signifie que le client a commencé le téléchargement du bloc et (●) signifie que le client a terminé le téléchargement du bloc.

Dans cette figure, v correspond à la valeur du variant dans chaque état du système. On pourra remarquer que la valeur de v décroît à chaque exécution d'un nouvel événement.

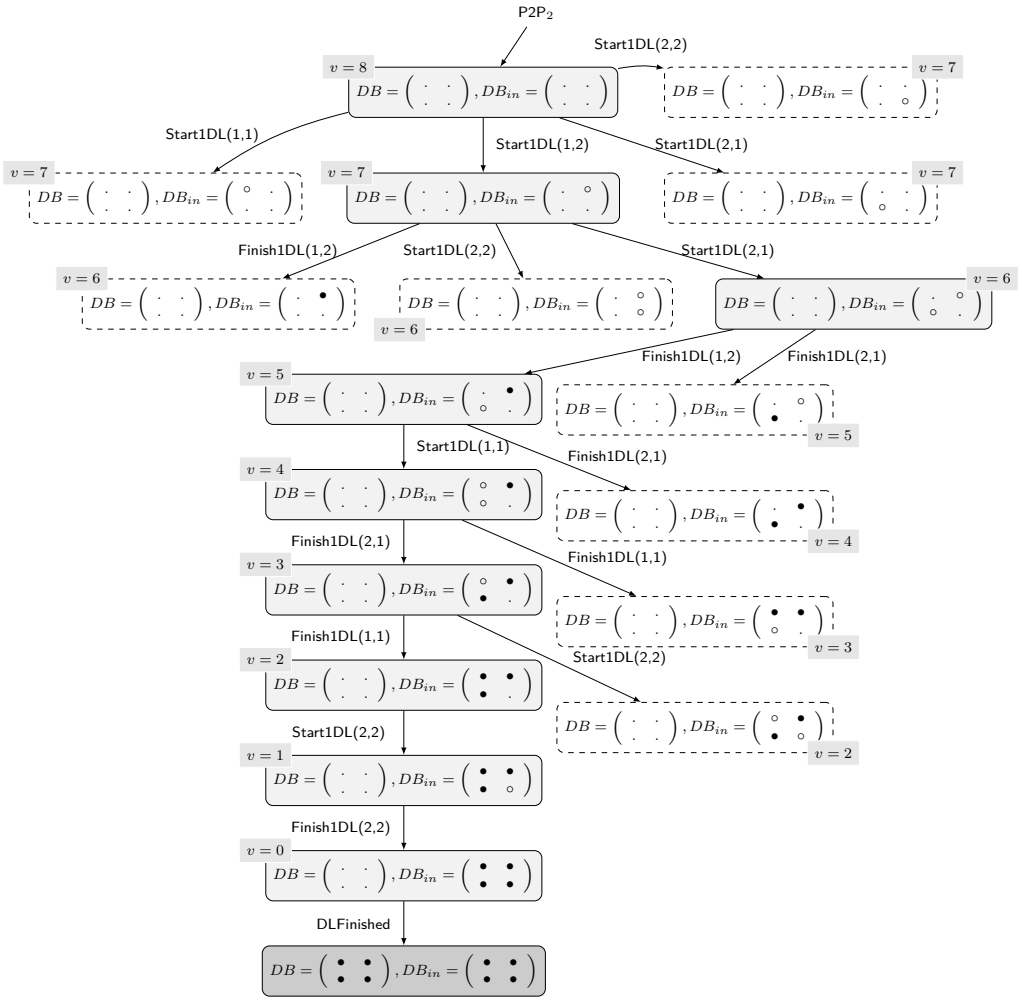

FIGURE 2.14 – Extrait du système de transitions correspondant à la machine $P2P_2$, avec N=2 et K=2

2.7 B Événementiel et probabilités

Comme présenté dès le début de ce manuscrit, le B événementiel n'introduit pas de concepts particuliers pour traiter les aspects probabilistes des systèmes. Ainsi, plusieurs travaux de recherche visant à marier B événementiel et probabilités ont été menés pour permettre la prise en compte des aspects probabilistes des systèmes en B événementiel.

À notre connaissance, le premier travail traitant de la problématique de l'ajout de probabilités au sein de B événementiel a été mené par Abrial et ses co-auteurs dans [78]. Dans cet article, les auteurs traitent les différentes possibilités pour intégrer des probabilités au sein de B événementiel. Ils suggèrent d'introduire les probabilités comme raffinement du non-déterminisme en B événementiel. Nous rappelons que le non-déterminisme en B événementiel peut avoir lieu à différents niveaux, comme dans le choix entre les événements activables dans le même état, dans le choix des valuations des paramètres d'un événement et dans les substitutions. Ce travail peut être considéré comme un ensemble de consignes ou de bonnes pratiques pour intégrer le raisonnement probabiliste en B événementiel. Les auteurs indiquent de plus que chaque extension de B événementiel pour supporter le raisonnement probabiliste doit respecter certaines exigences :

- Le B événementiel étendu doit rester simple et compréhensible ;
- Le B événementiel doit être générique, permettant ainsi la description d'une large classe de systèmes ;

- L'extension doit être facilement intégrable à la plateforme Rodin.

À notre connaissance, tous les travaux traitant l'intégration de probabilités dans le B événementiel se sont contentés de remplacer les substitutions non-déterministes par des substitutions probabilistes. Les extensions probabilistes au B événementiel existantes gardent ainsi du non-déterminisme qui apparaît alors au niveau du choix de l'événement qui sera exécuté parmi les événements activables et au niveau du choix des valuations des paramètres d'un événement.

Extension qualitative Le deuxième travail visant à intégrer le raisonnement probabiliste en B événementiel a été proposé par Hallerstede et Hoang dans [54]. Dans cet article, les auteurs introduisent un nouveau type de raisonnement en B événementiel appelé *Raisonnement probabiliste qualitatif*.

Ils proposent une extension qualitative du B événementiel dans laquelle les substitutions non-déterministes sous forme de prédicat sont remplacées par des substitutions probabilistes dite qualitatives. Les nouvelles substitutions s'écrivent de la façon suivante :

$$x :\oplus Q_x(\bar{t},\bar{v},x,x')$$

Dans cette substitution, une variable x prend une nouvelle valeur x' avec une valeur de probabilité qui n'est pas précisée mais qui doit être strictement positive.

Dans ce contexte, les auteurs étudient la convergence presque certaine d'un ensemble d'événements contenant des substitutions probabilistes qualitatives. Ils formalisent des conditions nécessaires pour prouver la convergence presque certaine de cet ensemble par des obligations de preuve. Pour cela, ils adaptent les obligations de preuve de convergence classique du B événementiel et introduisent de nouvelles obligations de preuve. Nous rappelons qu'en B événementiel, prouver qu'un ensemble d'événements converge revient à prouver que chaque événement de cet ensemble remplit les obligations de preuve evt/var/NAT et event/VAR présentées dans la section 2.4.3.

En particulier, l'obligation de preuve event/VAR assure pour chaque événement convergent que toutes les valuations des variables qui sont obtenues après l'exécution de son action décroissent la valeur du variant. Hoang et Hallerstede allègent cette contrainte formalisée par event/VAR : pour chaque événement contenant des substitutions probabilistes, ils imposent qu'au moins une valuation des variables obtenue après l'exécution de son action décroît la valeur du variant. Ainsi, il n'est plus nécessaire de prouver que toutes les valuations des variables obtenues après l'exécution de l'action de l'événement décroissent la valeur du variant, mais il suffit de trouver une seule valuation qui le fait décroitre. Cette condition est en effet suffisante à prouver que la probabilité de décroitre le variant est strictement positive. Pour un événement e_i, l'adaptation de cette obligation de preuve est alors donnée par :

evt/pVAR

$$\frac{I(\bar{v}) \wedge G_i(\bar{t},\bar{v})}{\vdash \exists\, \bar{v}'.\, S_i(\bar{t},\bar{v},\bar{v}') \wedge V(\bar{v}') < V(\bar{v})}$$

Notons que ceci implique que certaines valuations des variables obtenues après l'exécution de l'action de l'événement peuvent accroître la valeur du variant à partir du moment où au moins l'une d'entre elles le fait décroitre.

Pour empêcher l'augmentation infinie de la valeur du variant, Hallerstede et Hoang imposent une nouvelle contrainte : ils introduisent une borne maximale $\mathsf{U}(\bar{v})$ et ils imposent que la valeur du variant ne doit pas dépasser la valeur de cette borne maximale. Pour un événement e_i, cette contrainte est formalisée par une nouvelle obligation de preuve.

$$\frac{\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v})}{\mathsf{V}(\bar{v}) < \mathsf{U}(\bar{v})}\ \mathsf{evt/var/BOUND}$$

De plus, Hallerstede et Hoang imposent que chaque événement probabiliste convergent ne peut pas augmenter la valeur de la borne maximale. Pour un événement e_i, cette obligation de preuve est donnée formellement par :

$$\frac{\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v})}{\forall\, \bar{v}'.\, \mathsf{S}_i(\bar{t},\bar{v},\bar{v}') \Rightarrow \mathsf{U}(\bar{v}') < \mathsf{U}(\bar{v})}\ \mathsf{evt/bound/BOUND}$$

Hallerstede et Hoang ont alors adapté la sémantique ainsi que les obligations de preuve du B événementiel standard pour qu'ils puissent être appliqués dans leur extension. Cette extension est principalement utilisée pour faciliter la preuve de la convergence des nouveaux événements introduits durant le raffinement.

L'extension proposée par Hallerstede et Hoang a elle-même fait l'objet de plusieurs autres travaux de recherche. Dans [97], les mêmes auteurs étudient le raffinement des modèles B événementiel contenant des événements avec des substitutions probabilistes qualitatives. Ils développent aussi une extension de la plateforme Rodin pour permettre la spécification des événements contenant ces substitutions ainsi que les obligations de preuve associées. Dans [58], Hoang utilise une adaptation des obligations de preuve exprimées dans [54] pour prouver l'occurrence presque certaine de propriétés données.

Pour illustrer l'utilisation pratique de la proposition du raisonnement probabiliste qualitatif en B événementiel, d'autres travaux ont développé des études de cas ou exemples, en utilisant les éléments de cette extension, nous citons à titre d'exemples [98, 53].

Extension quantitative Adoptant la même démarche pour étendre B événementiel avec des probabilités (remplacer les substitutions non-déterministes par des substitutions probabilistes), d'autres travaux [91, 92, 93] ont proposé une extension probabiliste quantitative au B événementiel où les auteurs introduisent des substitutions probabilistes en précisant cette fois des valeurs de probabilité. Le premier travail de cette catégorie des travaux a été mené par Tarasyuk, Troubitsyna et Laibinis dans [91]. Dans cet article, les auteurs étudient l'évaluation de la fiabilité en B événementiel. Pour résoudre le problème du choix non-déterministe entre les événements activables, les auteurs traitent des modèles où, dans chaque valuation des variables, au plus un événement est activable. Considérant des systèmes décrits dans ces formalismes, ils proposent d'utiliser le *model*

checker Prism [70] pour vérifier la fiabilité [96] du système décrit dans leur extension. Nous notons que, dans cet article, les auteurs considèrent des modèles écrits directement dans ce nouveau formalisme, sans lien (raffinement) avec des modèles B événementiel standard. Le deuxième travail [92] (2010) a été mené par les mêmes auteurs. Ils proposent alors de remplacer (raffiner) les substitutions non-déterministes énumérés d'un modèle B événementiel standard par des substitutions probabilistes quantitatives afin de transformer ce modèle en modèle probabiliste. Une substitution probabiliste quantitative s'écrit :

$$\mathsf{x} \mid\oplus \mathsf{x}_1 \mathbin{@} p_1 \; ; \ldots ; \mathsf{x}_n \mathbin{@} p_n$$

La signification de cette substitution est comme suit : la variable x prend une des valeurs x_i ($1 \leq \mathsf{i} \leq \mathsf{n}$) avec une probabilité p_i. Ils imposent que la somme des valeurs des probabilités p_i doit être égale à 1. Le troisième travail [93] a été mené aussi par les mêmes auteurs. Ils introduisent la même substitution introduite dans [92] et ils expriment des modèles de systèmes cycliques en B événementiel avec des événements contenant des substitutions probabilistes. Ils utilisent alors ces modèles pour vérifier plusieurs propriétés des systèmes comme la fiabilité [96] et la réactivité [38, 94].

3

B événementiel probabiliste

3.1 Introduction

Nous rappelons que le principal objectif de notre travail consiste à combiner raisonnement probabiliste et B événementiel. Dans [78], Abrial *et al.* ont proposé différentes possibilités pour intégrer des probabilités à B événementiel : ils suggèrent de préciser le non-déterminisme par des probabilités. Nous rappelons que le non-déterminisme en B événementiel apparaît à trois niveaux :

1. entre les événements activables en même temps,
2. pour le choix des valeurs des paramètres d'un événement, et,
3. dans les substitutions non-déterministes.

Suite à [78], plusieurs autres travaux de recherche [58, 98, 53, 91, 92, 93] ont traité l'extension de B événementiel pour supporter des probabilités. Comme déjà mentionné dans la section 2.7 du chapitre 2, tous ces travaux ont principalement remplacé les substitutions non-déterministes par des susbtitutions probabilistes. Dans ce chapitre, nous présentons notre proposition d'extension de B événementiel, afin de supporter l'ensemble du raisonnement probabiliste. Dans notre proposition, nous précisons toutes les sources du non-déterminisme en B événementiel standard par des probabilités. Les modèles issus de cette extension seront appelés *Modèles B événementiel (purement) probabilistes*.

Dans la section 3.2, nous commençons par rappeler les sources de non-déterminisme en B événementiel et nous montrerons comment procéder pour les remplacer par des probabilités. Nous présenterons aussi la syntaxe de notre nouvelle proposition et la structure des machines probabilistes dans la section 3.3. Afin d'assurer l'exactitude des modèles B événementiel probabilistes, nous étudierons dans la section 3.4 la cohérence d'un modèle B événementiel probabiliste : un modèle B événementiel probabiliste correspond à un modèle B événementiel standard enrichi d'éléments de syntaxe spécifique pour décrire des probabilités. Ainsi, afin d'assurer la cohérence d'une machine B événementiel probabiliste, cette dernière doit satisfaire les obligations de preuve standard de B événementiel et elle doit également vérifier la correction des nouveaux éléments de syntaxe introduit. Finalement, nous présenterons dans la section 3.5 la sémantique opérationnelle d'un modèle B événementiel probabiliste et nous démontrerons qu'il s'agit bien d'une chaîne de Markov discrète.

3.2 Introduction des probabilités en B événementiel

Notre travail s'inscrit dans la suite des travaux traitant l'extension du B événementiel avec des probabilités. À notre connaissance, tous les travaux de recherche [78, 58, 98, 53, 91, 92, 93] qui ont étudié cette extension ont uniquement proposé de remplacer les substitutions non-déterministes par des substitutions probabilistes et ont gardé les autres sources de non-déterminisme intacts.

Dans notre travail, nous proposons de préciser toutes les sources de non-déterminisme en B événementiel par des probabilités rendant ainsi plus aisée la modélisation des systèmes purement probabilistes. Dans cette section, nous détaillons le passage du non-déterminisme vers les probabilités. Nous rappelons les différentes sources du non-déterminisme en B événementiel et nous introduisons pour chacune d'entre elles notre proposition pour la remplacer par des probabilités. Afin de bien expliquer les sources de non-déterminisme en B événementiel et comment nous avons procédé pour les remplacer par des probabilités, nous commençons par présenter un exemple décrivant une machine B événementiel non-déterministe. Dans un premier temps, nous rappelons où intervient le non-déterminisme dans cette machine. Dans un deuxième temps, nous précisons comment les remplacer par des probabilités.

3.2.1 Non-déterminisme en B événementiel : Exemple

MACHINE
Mch
VARIABLES
x

INVARIANT
x ∈ NAT
EVENTS

Init ≙
begin
x := 1
end

evt1 ≙
any t **where**
t ∈ NAT ∧ t < 3 ∧ x = 1
then
x :∈ {x+t,x+2t}
end

evt2 ≙
when x=1 **then**
x :| (1 ≤ x' ≤ 2)
end

END

FIGURE 3.1 – Machine Mch

La machine B événementiel donnée figure 3.1 contient une unique variable x qui est de type entier et qui est initialisée à 1. Elle contient deux événements non-déterministes evt1 et evt2.

- L'événement evt1 est activable lorsque x=1, il est paramétré par t, un entier tel que sa valeur est strictement inférieure à 3 : t peut donc être égal à 0, à 1 ou 2. L'action de evt1 contient une unique substitution énumérée non-déterministe qui affecte à x l'une des expressions x+t ou x+2t, le choix de l'expression affectée à x étant un choix non-déterministe ;
- L'événement evt2 est activable lorsque x=1, son action contient une substitution non-déterministe sous forme de prédicat qui affecte à x une nouvelle valeur x' telle que le prédicat 1≤x'≤2 est satisfait : x peut donc prendre la valeur 1 ou 2.

Nous présentons dans la figure 3.2 le système de transitions illustrant la sémantique opérationnelle de cette machine. Nous détaillons dans la figure 3.3 la construction détaillée du système de transitions précédent.

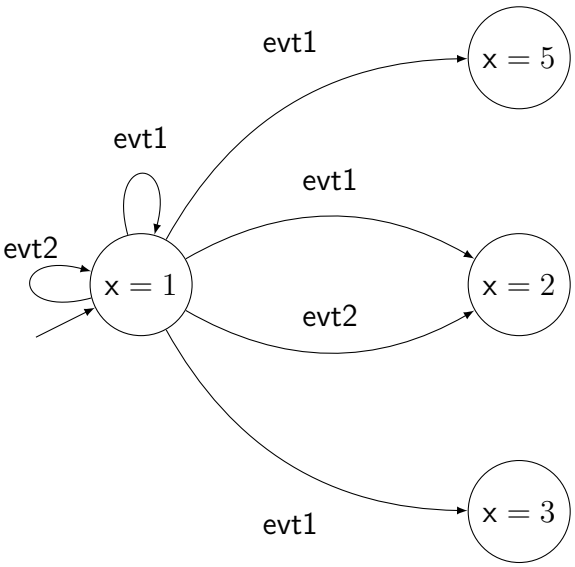

FIGURE 3.2 – Sémantique opérationnelle de la machine Mch

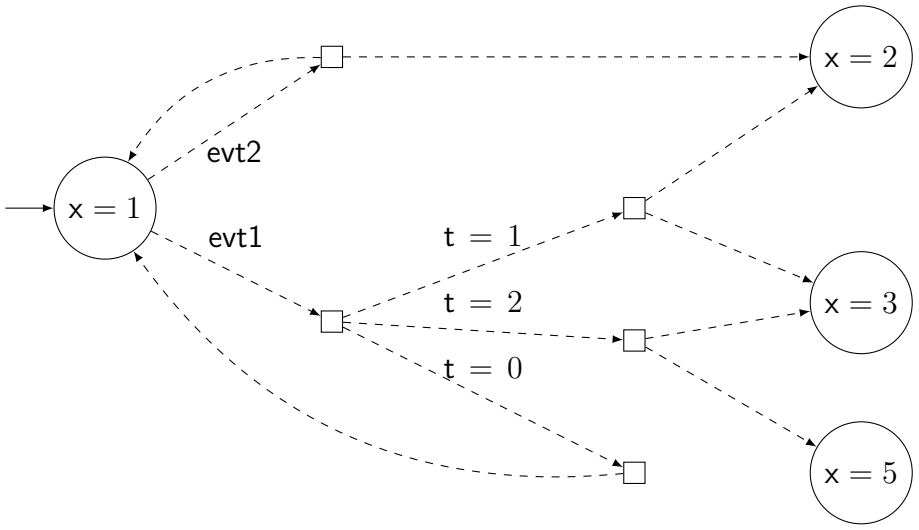

FIGURE 3.3 – Construction détaillée de la sémantique opérationnelle de la machine Mch

Dans cette construction, les transitions en pointillé sont des transitions “intermédiaires” qui décomposent les choix non-déterministes à l'intérieur de l'événement. Les états □ représentent des états “intermédiaires”, où ont lieu des choix non-déterministes.

Dans l'état (x=1), evt1 et evt2 sont activables, l'un de ces événements est choisi pour être exécuté d'une manière non-déterministe : si evt1 est choisi, une valeur pour le paramètre t doit également être choisie.

- Considérons le choix de la valeur 0 pour le paramètre t, une des expressions x+t ou x+2t doit alors être choisie d'une manière non-déterministe pour être affectée à x et chacune des expressions va prendre la valeur 1 qui sera affectée à x,
- Considérons le choix de la valeur 1 pour le paramètre t, une des expressions x+t ou x+2t doit alors être choisie d'une manière non-déterministe pour être affectée à x. Si x+t est choisie, alors x prend la valeur 2, si x+2t est choisie, alors x prend la valeur 3,
- Considérons le choix de la valeur 2 pour le paramètre t, une des expressions x+t ou x+2t doit

alors être choisie d'une manière non-déterministe pour être affectée à x. Si x+t est choisie, alors x prend la valeur 3, si x+2t est choisie, alors x prend la valeur 5.

Si l'événement evt2 est choisi, alors visiblement x' va prendre soit la valeur 1 ou 2 et une de ces valeurs va être affectée à la variable x.

3.2.2 Introduction des probabilités dans la machine Mch

Nous allons maintenant indiquer comment introduire des probabilités à chaque place où apparaissait du non-déterminisme.

Annotation des événements par des poids Nous proposons d'annoter l'événement evt1 par un poids x+1 et l'événement evt2 par un poids x+2.

- L'événement evt1 étant annoté par un poids x+1, cette expression s'évalue à 2 dans l'état (x=1) ;
- L'événement evt2 étant annoté par un poids x+2, cette expression s'évalue à 3 dans l'état (x=1).

Comme dans le cas standard, dans l'état (x=1), l'un des événements evt1 ou evt2 est choisi pour être exécuté. L'événement evt1 est exécuté avec une probabilité $\frac{2}{5}$ qui correspond au ratio de son poids divisé par la somme des poids des événements activables. De la même manière, l'événement evt2 est exécuté avec une probabilité $\frac{3}{5}$. Le choix entre ces deux événements est ainsi un choix probabiliste et pas non-déterministe comme le cas standard, comme illustré dans la figure 3.4.

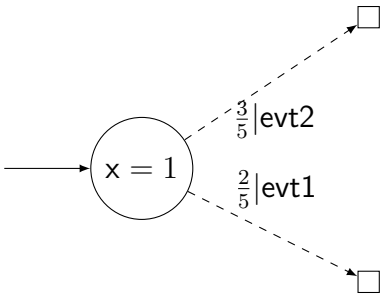

FIGURE 3.4 – Choix probabiliste de l'événement à exécuter

Choix uniforme des valeurs du paramètre t L'événement evt1 possède un paramètre t qui prend une valeur telle que $0 \leq t < 3$ ($t \in \{0,1,2\}$). Nous proposons que le choix de la valeur de paramètre soit effectué d'une manière uniforme. Ainsi, dans l'état intermédiaire obtenu après le choix de l'événement evt1, une valeur du paramètre t est choisie dans l'ensemble $\{0,1,2\}$ avec une probabilité $\frac{1}{3}$ pour chaque valeur. Voir figure 3.5.

Substitutions probabilistes dans la machine PMch L'événement evt1 possède une substitution énumérée non-déterministe

$$x :\in \{x+t, x+2t\}$$

Nous proposons de remplacer cette substitution par une substitution probabiliste énumérée

$$x := x+t \ @\ 1/3 \oplus x+2t \ @\ 2/3$$

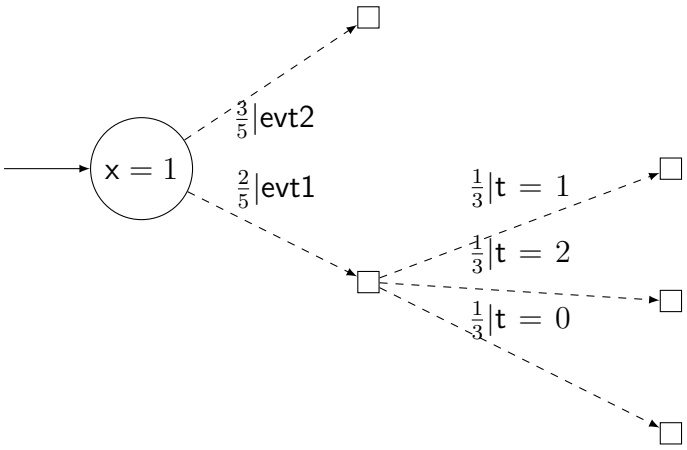

FIGURE 3.5 – Choix uniforme de la valeur du paramètre t

Nous remarquons que dans l'état □ obtenu après le choix de la valeur 2 pour le paramètre t de l'événement evt_1, l'action de cet événement mène le système vers l'état (x=3) si l'expression x+t est affectée à x ou vers l'état (x=5) si l'expression (x+2t) est choisie pour être affectée à x.

De même, dans evt2, la substitution non-déterministe sous forme de prédicat est remplacée par une substitution probabiliste sous forme de prédicat

$$x :\oplus (1 \leq x' \leq 2)$$

Nous présentons dans la figure 3.6 une construction détaillée de la sémantique de la machine PMch.

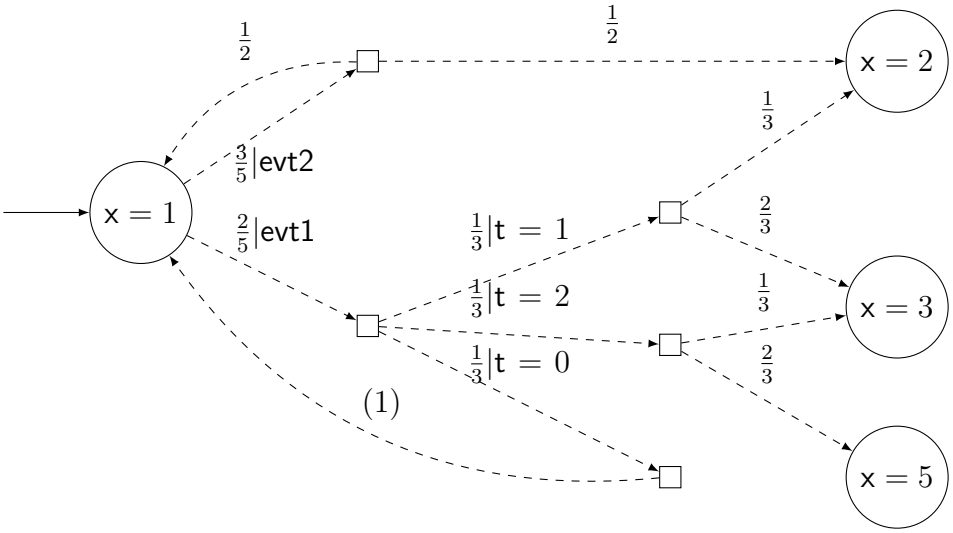

FIGURE 3.6 – Construction détaillée de la sémantique de la machine PMch

Nous présentons dans la figure 3.7 la chaîne de Markov correspondante à la machine PMch. Les probabilités des transitions de cette chaîne correspondent au produit du choix de l'événement par la probabilité du choix des valeurs des paramétres multiplié par la probabilité du choix des valeurs des variables. Par exemple, pour la transition de l'état (x=1) vers l'état (x=2) par l'intermédiaire de l'événement evt1, la probabilité de cette transition est $\frac{2}{5} \times \frac{1}{3} \times \frac{1}{3}$. Cette valeur de probabilité correspond au produit de la probabilité du choix de l'événement $\frac{2}{5} = \frac{2}{2+3}$ par la probabilité de choisir la valeur 1 pour le paramètre t par la probabilité d'affectation de la valeur 2 pour la variable x.

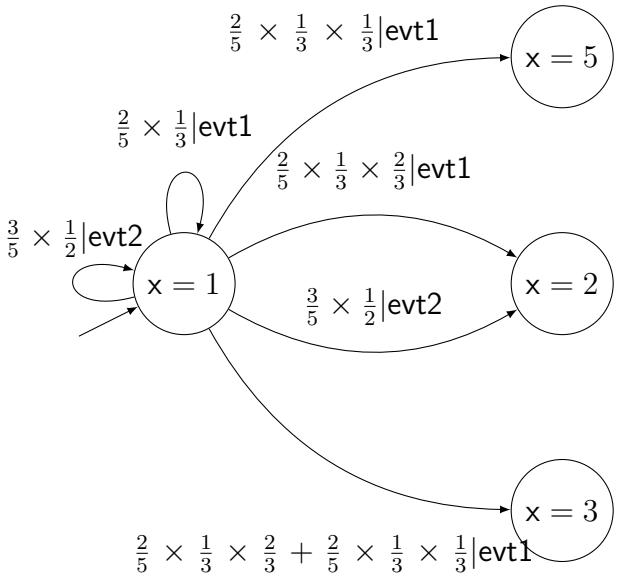

FIGURE 3.7 – Chaîne de Markov résultante

3.3 B événementiel probabiliste

Formalisons maintenant les éléments introduits précédemment.

3.3.1 Choix de l'événement à exécuter

Le comportement dynamique d'un système est décrit en B événementiel par des événements. D'une manière générale, plusieurs événements peuvent être activables dans la même valuation des variables, l'événement qui sera exécuté parmi ces événements est alors choisi d'une manière non-déterministe. Nous avons étudié le remplacement de ce choix non-déterministe par un choix probabiliste.

Une première solution consiste à annoter chaque événement par une valeur de probabilité. Cependant, en B événementiel, l'ensemble des événements activables peut changer d'un état à l'autre. Ainsi, la probabilité de chaque événement devrait également changer d'un état à l'autre et il est possible que la somme des probabilités des événements activables dans un état donné soit différente de un. Ceci aurait comme conséquence de produire une distribution de probabilité mal-définie. Cela nous amènerait à normaliser ces valeurs de probabilité dans tous les états du système, ce qui peut être compliqué. Une deuxième solution consiste à annoter chaque événement par un poids. Ainsi, dans un état où plusieurs événements sont activés, la probabilité qu'un événement parmi ces événements soit exécuté correspond au ratio du poids de cet événement par la somme des poids de tous les événements activés dans ce même état.

Dans notre proposition, nous avons opté pour cette solution : chaque événement e_i est annoté par un poids $\mathsf{W}_i(\bar{v})$ introduit par le mot clé **weight**. Pour plus d'expressivité, le poids $\mathsf{W}_i(\bar{v})$ de chaque événement e_i est une expression qui dépenderait des variables $\bar{v}$ de la machine B et qui sera évaluée dans chaque état du système, ce poids s'évalue à une valeur du type entier. Ce poids (et donc la probabilité associée) d'exécuter un événement particulier peut donc évoluer avec la progression du système.

L'annotation d'un événement par un poids $\mathsf{W}_i(\bar{v})$ ajoute des contraintes sur l'activation de l'événement : en B événementiel standard, un événement est activable dans un état si et seulement si les valeurs des variables dans cet état satisfont la garde. Dans notre proposition, puisqu'un événement est annoté par un poids, il est activable dans un état si et seulement si les valeurs des

variables dans cet état satisfont la garde et si l'évaluation de l'expression du poids dans cet état donne un entier strictement positif. En effet, si l'expression $W_i(\bar{v})$ de poids d'un événement dans un état donné s'évalue à zéro, alors la probabilité d'exécuter cet événement dans cet état est zéro et l'événement ne peut pas être exécuté.

3.3.2 Choix uniforme des valeurs des paramètres

Comme présenté dans la section 2.2.3 du chapitre 2, un événement en B événementiel peut posséder des paramètres. Les types correspondant aux différents paramètres sont précisés dans la garde de l'événement. Dans chaque état du système, les valeurs des paramètres sont choisies de telle sorte que la garde soit satisfaite lorsque qu'elle est évaluée avec ces valeurs de paramètres et les valeurs des variables dans cet état. En B événementiel standard, lorsqu'il y a plusieurs valeurs possibles des paramètres qui peuvent satisfaire la garde de l'événement, une de ces valeurs est choisie d'une manière non-déterministe. Afin de lever ce non-déterminisme, nous proposons de remplacer ce choix non-déterministe par un choix uniforme discret parmi toutes les valeurs possibles des paramètres qui satisfont la garde de l'événement dans un état donné.

Pour des raisons de simplicité, nous avons opté pour un choix uniforme discret, mais ce choix pourrait être remplacé par d'autres distributions discrètes. Ce choix influence le calcul des valeurs des probabilités lors de l'exécution de l'événement, nous détaillons cet aspect dans la section 3.5. L'avantage de ce choix est qu'il ne surcharge pas les notations, puisqu'il n'y a rien à préciser de spécial.

3.3.3 Les substitutions probabilistes

Un événement en B événementiel possède une action. Cette action est constituée de plusieurs substitutions. Comme présenté dans la section 2.2.4 du chapitre 2, certaines substitutions sont non-déterministes. Ces substitutions sont de deux types : *les substitutions énumérées non-déterministes* et *les substitutions non-déterministes sous forme de prédicat*. Nous montrons dans ce qui suit comment remplacer ces substitutions par des substitutions probabilistes.

Substitution probabiliste sous forme de prédicat En B événementiel, une substitution non-déterministe sous forme de prédicat s'écrit :

$$x :| Q_x(\bar{t},\bar{v},x,x')$$

L'interprétation de cette substitution est comme suit : la variable x prend une nouvelle valeur x' parmi les valeurs possibles qui satisfont le prédicat $Q_x(\bar{t},\bar{v},x,x')$. En B événementiel standard, le choix d'une nouvelle valeur x' parmi toutes les valeurs possibles se fait de manière non-déterministe. Nous proposons de remplacer cette substitution par une substitution probabiliste qui s'écrit :

$$x :\oplus Q_x(\bar{t},\bar{v},x,x')$$

L'interprétation de cette susbtitution est comme suit : au lieu de choisir une nouvelle valeur x' d'une manière non-déterministe, nous choisissons cette nouvelle valeur x' d'une manière uniforme sur l'ensemble des valeurs x' qui satisfont le prédicat $Q_x(\bar{t},\bar{v},x,x')$.

Comme pour les paramètres et pour des raisons de simplicité, nous imposons que la distribution sur l'ensemble des valeurs x' qui satisfont $Q_x(\bar{t},\bar{v},x,x')$ soit uniforme. Ce choix de distribution uniforme discrète est fait pour des raisons de simplicité, nous pourrions remplacer cette distribution par un autre type de distribution discrète ou continue. Ce choix de type de substitution influence le calcul des valeurs de probabilité de l'exécution d'un événement. L'avantage est là encore de ne pas surcharger inutilement les notations B événementielles.

Substitution énumérée probabiliste En B événementiel une substitution énumérée non-déterministe s'écrit :

$$x :\in \{E_1(\bar{t},\bar{v}) \ldots E_m(\bar{t},\bar{v})\}$$

L'interprétation est comme suit : une expression $E_i(\bar{t},\bar{v})$) est choisie parmi toutes les expressions $E_1 \ldots E_m$ pour être affectée à la variable x. Le choix de cette expression se fait d'une manière non-déterministe.

Nous proposons de remplacer chaque substitution énumérée non-déterministe par une substitution énumérée probabiliste qui s'écrit :

$$x := E_1(\bar{t},\bar{v}) \text{ @ } p_1 \oplus \ldots \oplus E_m(\bar{t},\bar{v}) \text{ @ } p_m$$

L'interprétation de cette nouvelle substitution est comme suit, une expression $E_i(\bar{t},\bar{v})$ ($1 \leq i \leq m$) est choisie pour être affectée à la variable x avec une probabilité p_i. Cette valeur de probabilité p_i est une constante de type rationnel.

Note 2 *Les rationnels ne sont pas nativement gérés en B événementiel. Une solution que nous adoptons consiste à définir en B événementiel un type rationnel. Par exemple, nous pouvons définir une théorie des rationnels dans la plateforme Rodin en utilisant le plugin* Theory *de Butler et Maamria [37] permettant d'étendre Rodin avec de nouveaux concepts mathématiques.*

Plus généralement, nous aurions pu mettre des expressions en fonction des variables et des paramètres pour les valeurs de probabilité p_i mais nous avons opté pour ce choix de contantes pour des raisons de simplicité.

Pour définir une distribution discrète correcte sur l'ensemble des expressions qui peuvent être affectées à x, l'ensemble des expressions doit être fini et chaque valeur de probabilité p_i doît être un rationnel strictement positif et inférieur ou égal à 1. De plus, la somme de toutes les valeurs de probabilité p_i doît être égale à 1.

Prédicat avant-après d'une action probabiliste Comme en B événementiel standard, la sémantique d'une substitution probabiliste est décrite par un prédicat avant-après qui spécifie la relation entre la valeur d'une variable avant (x) et après (x') l'application de la substitution. Le prédicat avant-après de chacune de deux nouvelles substitutions probabilistes ajoutées correspond exactement au prédicat avant-après de la substitution non-déterministe qui lui correspond :

- Le prédicat avant-après correspondant à une substitution énumérée probabiliste $x := E_1(\bar{t},\bar{v}) \text{ @ } p_1 \oplus \ldots \oplus E_m(\bar{t},\bar{v}) \text{ @ } p_m$ est donné par $Q_x(\bar{t},\bar{v},x,x') = x' \in \{E_1(\bar{t},\bar{v}) \ldots E_m(\bar{t},\bar{v})\}$.
- Le prédicat avant-après correspondant à une substitution probabiliste sous forme de prédicat $x :\oplus Q_x(\bar{t},\bar{v},x,x')$ est donné par le prédicat $Q_x(\bar{t},\bar{v},x,x')$.

Une action probabiliste peut être aussi décrite par un prédicat avant-après. Ce prédicat s'exprime de la même manière que son correspondant non-déterministe. Supposons que les variables $x_1 \ldots x_i$ sont les variables modifiées par l'action $SP_i(\bar{t},\bar{v})$ et que les variables $x_{i+1} \ldots x_n$ sont non touchées. Supposons que $Q_{x_1}(\bar{t},\bar{v},x_1,x_1') \ldots Q_{x_i}(\bar{t},\bar{v},x_i,x_i')$ sont les prédicats avant-après correspondant aux substitutions $S_1 \ldots S_i$, le prédicat avant-après de l'action $SP_i(\bar{t},\bar{v})$ de cet événement est alors exprimé par :

$$SP_i(\bar{t},\bar{v},\bar{v}') \mathrel{\widehat{=}} Q_{x_1}(\bar{t},\bar{x},x_1,x_1') \wedge \ldots \wedge Q_{x_i}(\bar{t},\bar{x},x_i,x_i') \wedge (x_{i+1}'=x_{i+1}) \wedge \ldots (x_n'=x_n)$$

Le prédicat avant-après d'une action probabiliste est identique au prédicat avant-après de l'action correspondante. Ceci facilite la génération des obligations de preuve dans l'outil Rodin puisque c'est le même prédicat qui est utilisé. Ce qui montre bien que notre extension probabiliste au B événementiel est simple et facilement intégrable à Rodin.

3.3.4 Machine B événementiel probabiliste

Remplacer tous les choix non-déterministes en B événementiel par des choix probabilistes implique des modifications sur la syntaxe des événements et des machines. On parle dans ce cas de machines B événementiel probabilistes et d'événements probabilistes. Nous présentons dans la figure 3.8 la forme d'un événement probabiliste. Dans cette figure, e_i représente le nom de l'événement, $\mathsf{W}_i(\bar{v})$ représente le poids de l'événement, $\mathsf{G}_i(\bar{t},\bar{v})$ représente la garde de l'événement et $\mathsf{SP}_i(\bar{t},\bar{v})$ représente l'action probabiliste de l'événement. Une action probabiliste est composée uniquement de substitutions déterministes et de substitutions probabilistes.

$\mathsf{e}_i \ \widehat{=}$
weight
$\mathsf{W}_i(\bar{v})$
any
$\bar{t}$
where
$\mathsf{G}_i(\bar{t},\bar{v})$
then
$\mathsf{SP}_i(\bar{t},\bar{v})$
end

FIGURE 3.8 – Forme d'un événement probabiliste

La structure d'une machine B événementiel probabiliste est présentée dans la figure 3.9. Une machine B événementiel probabiliste a la même structure qu'une machine B événementiel standard. La seule différence concerne l'ensemble des événements où dans une machine probabiliste, l'ensemble des événements (PEvts) ne peut contenir que des événements probabilistes. Pour une utilisation plus simple des machines probabilistes, nous dénotons dans la suite de ce chapitre une machine probabiliste par un tuple pmch=($\bar{v}$,$\overline{\mathsf{ctx}}$,I($\bar{v}$),Init,PEvts,V($\bar{v}$)). Nous notons que comme c'est le cas pour les machines B événementiel standard, l'événement d'initialisation d'une machine B événementiel probabiliste doit être déterministe.

MACHINE
pmch
SEES
$\overline{\mathsf{ctx}}_1$,$\overline{\mathsf{ctx}}_2$,...
VARIABLES
$\bar{v}$
INVARIANT
I($\bar{v}$)
VARIANT
V($\bar{v}$)
EVENTS
PEvts
END

FIGURE 3.9 – Structure d'une machine probabiliste

MACHINE
P2P$_p$
VARIABLES
DB
DBin
INVARIANTS
DB ∈ 1..N × 1..K → {empty,finished} ∧ DBin ∈ 1..N × 1..K → {empty,incoming,finished} ∧
∀ c . (c ∈ 1..N ⇒ card({b | b ∈ 1..K ∧ DBin(c↦b) = incoming}) ≤ 1)
VARIANT
2 × N × K − 2 × card({c↦b | c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = finished})
− card({c↦b | c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = incoming})
EVENTS
Init ≙
begin
DB := (1..N × 1..K) × {empty} ||
DBin := (1..N × 1..K) × {empty}
end
DLFinished ≙
weight N × K
when
DB = (1..N × 1..K) × {empty} ∧ DBin = (1..N × 1..K) × {finished}
then
DB := DBin
end
Start1DL ≙
weight
N × K − card({c↦b | c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = incoming})
any c, b **where**
c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = empty ∧
card({k | k ∈ 1..K ∧ DBin(c↦k) = incoming})=0
then
DBin(c↦b) := incoming
end
Finish1DL ≙
weight
card({c↦b | c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = finished}) +1
any c, b **where** c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = incoming
then
DBin(c↦b) := finished
end
FailureDL ≙
weight
N × K − card({c↦b |c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = finished})
any c, b **where** c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = incoming
then
DBin(c↦b) := empty @4/10 ⊕ incoming @6/10
end
END

FIGURE 3.10 – Une description probabiliste du protocole P2P en B événementiel probabiliste

3.3.5 Étude de cas : protocole pair à pair en B événementiel probabiliste

Afin d'illustrer notre proposition de B événementiel probabiliste et afin d'expliquer les nouveaux éléments de syntaxe introduits précédemment, nous reprenons ici l'étude de cas du protocole pair à pair présenté dans la section 2.5 du chapitre 2. Rappelons que ce cas d'étude représente un ensemble de N clients qui essayent de télécharger un fichier partitionné en K blocs. Le protocole se termine lorsque tous les clients ont réussi à télécharger tous les blocs.

Nous partons de la machine B événementiel $P2P_3$ présentée dans la figure 2.10 et nous en donnons une version probabiliste : la machine $P2P_p$ présentée dans la figure 3.10. Cette machine possède les mêmes variables, le même invariant, le même variant ainsi que les mêmes événements que la machine $P2P_3$.

L'événement Start1DL représente le début du téléchargement d'un bloc par un client, l'événement Finish1DL représente la finalisation du téléchargement d'un bloc par un client et l'événement FailureDL représente l'occurence d'erreur durant le téléchargement d'un bloc par un client.

Dans la description informelle de ce protocole, le risque d'occurence d'erreur durant le téléchargement d'un bloc par un client diminue avec l'augmentation du nombre de téléchargements effectués. Dans la description en B événementiel standard présentée dans la machine $P2P_3$, cette contrainte ne peut pas être prise en compte.

- Dans $P2P_p$, nous annotons l'événement Finish1DL par le poids suivant :

$$\mathsf{card}(\{\mathsf{n} \mapsto \mathsf{b} \mid \mathsf{n} \in 1..\mathsf{N} \land \mathsf{b} \in 1..\mathsf{K} \land \mathsf{DBin}(\ \mathsf{n} \mapsto \mathsf{b}\ ) = \mathsf{finished}\}) + 1$$

  Cette expression du poids représente le nombre des téléchargements terminés par tous les clients du protocole, elle augmente donc à chaque exécution de Finish1DL.

- Nous annotons aussi FailureDL par le poids suivant :

$$\mathsf{N} \times \mathsf{K} - \mathsf{card}(\{\mathsf{n} \mapsto \mathsf{b} \mid \mathsf{n} \in 1..\mathsf{N} \land \mathsf{b} \in 1..\mathsf{K} \land \mathsf{DBin}(\ \mathsf{n} \mapsto \mathsf{b}\ ) = \mathsf{finished}\})$$

  Cette expression de poids représente le nombre de téléchargements restant à réaliser.

Nous remarquons qu'à chaque fois qu'un bloc est téléchargé par un client, le poids de Finish1DL va augmenter alors que le poids de FailureDL va diminuer, ce qui correspond à diminuer l'occurence d'erreur en fonction du nombre de téléchargements effectués.

Pour l'événement Start1DL qui représente le déclenchement du téléchargement d'un bloc b par un client c, nous l'annotons par le poids suivant :

$$\mathsf{N} \times \mathsf{K} - \mathsf{card}(\{\mathsf{n} \mapsto \mathsf{b} \mid \mathsf{n} \in 1..\mathsf{N} \land \mathsf{b} \in 1..\mathsf{K} \land \mathsf{DBin}(\ \mathsf{n} \mapsto \mathsf{b}\ ) = \mathsf{finished}\})$$

Le poids de l'événement Start1DL modélise le fait que la probabilité de commencer un nouveau téléchargement diminue avec l'augmentation du nombre de blocs téléchargés par tous les clients. On voit ici l'importance d'autoriser des expressions pour les poids des événements et pas des constantes. C'est l'expressivité des poids des événements qui nous permettent d'exprimer facilement ce type de contraintes issues de la description informelle.

Nous rappelons que deux sortes d'erreur peuvent arriver : la première correspond à l'interruption du téléchargement alors que la deuxième décrit l'impossibilité du téléchargement à un instant donné. La probabilité d'interruption de téléchargement est fixée à $\frac{4}{10}$ alors que la probabilité d'impossibilité d'effectuer le téléchargement est fixée à $\frac{6}{10}$. Ces valeurs sont difficilement spécifiables en B événementiel classique. En B événementiel probabiliste, puisque nous introduisons les substitutions énumérées probabilistes, nous utilisons la substitution $(\mathsf{DBin}(\ \mathsf{c} \mapsto \mathsf{b}\ ) := \mathsf{empty}\ @\ 4/10 \oplus \mathsf{incoming}\ @\ 6/10)$ afin de spécifier ces mesures.

3.4 Cohérence d'une machine B évenementiel probabiliste

Comme dans le cas du B événementiel standard, la cohérence d'une machine B événementiel probabiliste est définie par des obligations de preuve. Dans cette section, nous étudions la cohérence de ces machines, nous introduisons des nouvelles obligations de preuve spécifiques au B événementiel probabiliste et nous adaptons les obligations de preuve standards pour qu'elles puissent être appliquées sur une machine B événementiel probabiliste.

3.4.1 Nouvelles obligations de preuve

Nous présentons dans ce qui suit les nouvelles obligations de preuve spécifiques à une machine B événementiel probabiliste.

Le poids d'un événement probabiliste doit s'évaluer en une valeur de type entier Pour des raisons de simplicité, pour chaque événement probabiliste e_i telle que sa garde $\mathsf{G}_i(\bar{t},\bar{v})$ est vraie, l'expression de son poids $\mathsf{W}_i(\bar{v})$ doit s'évaluer en une valeur de type entier. Formellement :

$$\frac{\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \vdash \mathsf{W}_i(\bar{v}) \in \mathsf{NAT}}{\text{evt/WGHT/NAT}}$$

Finitude de l'ensemble des valeurs des paramètres d'un événement Afin de pouvoir définir une distribution discrète uniforme sur l'ensemble des valeurs des paramètres d'un événement probabiliste e_i qui satisfont la garde, cet ensemble doit être fini. Formellement :

$$\frac{\mathsf{I}(\bar{v}) \wedge \mathsf{W}(\bar{v}) > 0 \vdash \mathsf{finite}\ (\{\bar{t} \mid \mathsf{G}_i(\bar{t},\bar{v})\})}{\text{evt/param/pWD}}$$

Bonne définition et faisabilité des substitutions énumérées probabilistes Pour chaque substitution énumérée probabiliste, on doit s'assurer que les valeurs des probabilités $\mathsf{p}_1 \ldots \mathsf{p}_n$ définissent une distribution de probabilité correcte. Rappelant qu'une substitution énumérée probabiliste s'écrit :

$$\mathsf{x} := \mathsf{E}_1(\bar{t},\bar{v})\ @\ \mathsf{p}_1 \oplus \ldots \oplus \mathsf{E}_m(\bar{t},\bar{v})\ @\ \mathsf{p}_m$$

Cette contrainte est formellement représentée par deux obligations de preuve :

1. Les valeurs de probabilité utilisées dans les substitutions énumérées probabilistes doîvent être strictement positives et inférieures ou égales à 1. Formellement :

$$\frac{}{\vdash 0 < p_i \leq 1}\ \text{evt/assign/pWD1}$$

2. Dans chaque substitution énumérée probabiliste, la somme des valeurs de probabililté utilisées doît être égale à 1. Formellement :

$$\frac{}{\vdash p_1 + \ldots + p_n = 1}\ \text{evt/assign/pWD2}$$

Nous remarquons que la faisabilité des substitutions énumérées probabilistes est triviale, il suffit qu'il existe au moins une seule expression $E_i(\bar{t},\bar{v})$ bien définie syntaxiquement.

Bonne définition et faisabilité des substitutions probabilistes sous forme de prédicat Pour chaque substitution probabiliste sous forme de prédicat, afin de définir une distribution uniforme discrète sur l'ensemble des valeurs d'une variable x qui satisfont le prédicat $Q_x(\bar{t},\bar{v},x,x')$ de la substitution probabiliste, cet ensemble doît être fini et non vide.

Rappelant qu'une substitution probabiliste sous forme de prédicat s'écrit :

$$x :\oplus\ Q_x(\bar{t},\bar{v},x,x')$$

L'obligation de preuve de faisabilité des substitutions sous forme de prédicat standard (voir section 2.3) qui garantit que l'ensemble $\{x' \mid Q_x(\bar{t},\bar{v},x,x')\}$ n'est pas vide.

La contrainte sur la finitude de l'ensemble des valeurs x' est représentée par une obligation de preuve qui est formellement présentée comme suit :

$$\frac{I(\bar{v}) \wedge G_i(\bar{t},\bar{v}) \wedge W_i(\bar{v}) > 0}{\vdash\ \text{finite}\ (\{x' \mid Q_x(\bar{t},\bar{v},x,x')\})}\ \text{evt/assign/pWD3}$$

Nous avons choisi la distribution uniforme discrète mais cette distribution peut être remplacée par une distribution continue ou discrète. Ce choix est fait pour des raisons de simplicité. Le changement du type de distribution induirait des modifications sur le calcul de probabililté d'exécution d'un événement, nous détaillerons cet aspect dans la section 3.5.

3.4.2 Adaptation des obligations de preuve standard

Une machine B événementiel probabiliste correspond à une machine B événementiel standard enrichie d'éléments de syntaxe spécifiques pour qu'on puisse exprimer des probabilités.

Lorsque il s'agit des obligations de preuve standard, la spécifité des événements probabilistes par rapport aux événements standards concerne leur condition d'activation. En B événementiel, un événement est activé si sa garde est satisfaite. En B événementiel probabiliste, il n'est pas suffisant que la garde de l'événement soit satisfaite, il faut en plus que le poids de l'événement soit évalué en un entier strictement positif. Ainsi, nous modifions toutes les obligations de preuve standard en ajoutant aux hypothèses de chaque obligation de preuve le prédicat $\mathsf{W}_i(\bar{v})>0$ spécifiant que le poids de l'événement est strictement positif.

Faisabilité des événements L'objectif de cette obligation de preuve est de s'assurer que l'action de chaque événement probabiliste est faisable. Formellement :

evt/pFIS

$$\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \wedge \mathsf{W}_i(\bar{v}) > 0 \vdash \exists\, \bar{v}'.\ \mathsf{SP}_i(\bar{t},\bar{v},\bar{v}')$$

Préservation de l'invariant Dans le cas probabiliste, cette obligation spécifie que l'exécution de chaque événement probabiliste préserve l'invariant. Formellement :

evt/pINV

$$\mathsf{I}(\bar{v}) \wedge \mathsf{G}_i(\bar{t},\bar{v}) \wedge \mathsf{W}_i(\bar{v}) > 0 \wedge \mathsf{SP}_i(\bar{t},\bar{v},\bar{v}') \vdash \mathsf{I}(\bar{v}')$$

Cas particulier de l'initialisation Puisque l'événement d'initialisation dans une machine B événementiel probabiliste est déterministe, et donc n'a pas de poids, nous appliquons la même obligation de preuve que celle présentée dans la section 2.3 du chapitre 2.

Absence de blocage Cette obligation de preuve assure que pour toutes les valeurs possibles des variables de la machine qui respectent l'invariant, au moins un événement est activé. En B événementiel standard, vu qu'un événement n'est activé que lorsque sa garde est satisfaite, l'hypothèse de cette obligation de preuve est l'invariant, la conclusion est la disjonction des conditions d'activation de tous les événements de la machine. Nous rappelons qu'en B événementiel, un événement

n'est activé dans une valuation des variables que lorsque sa garde est satisfaite dans cette valuation et l'expression de son poids s'évalue en un entier strictement positif dans cette valuation des variables. La condition d'activation de chaque événement probabiliste contient ainsi une prédicat spécifiant que son poids est strictement positif. Formellement :

machine/pDLF
$$\frac{}{\mathsf{I}(\bar{v})\vdash(\mathsf{G}_1(\bar{t},\bar{v})\ \wedge\ \mathsf{W}_1(\bar{v})>0))\vee\ldots\vee(\mathsf{G}_n(\bar{t},\bar{v})\ \wedge\ \mathsf{W}_n(\bar{v})>0)}$$

Nous notons que cette obligation de preuve est optionnelle en B événementiel.

3.4.3 Étude de cas : obligations de preuve appliquées au protocole pair à pair

Considérons la machine $\mathsf{P2P}_p$ présentée dans la figure 3.10. Visiblement, les expressions des poids des événements s'évaluent tous à des entiers naturels. Ainsi, chacun des événements respecte l'obligation de preuve (evt/WGHT/NAT). Les événements Star1DL et FailureDL possédent deux paramètres qui sont un client $\mathsf{c} \in 1..\mathsf{N}$ et un bloc $\mathsf{k}\in 1..\mathsf{K}$. Nous déduisons ainsi que l'obligation de preuve sur la finitude des valeurs des paramètres (evt/param/pWD) est bien respectée par chacun de ces événements. L'événement FailureDL contient une substitution non-déterministe énumérée $\mathsf{DBin}(\mathsf{c}\mapsto\mathsf{b}) := \mathsf{empty}\ @4/10 \oplus \mathsf{incoming}\ @6/10$. Visiblement, chacune des valeurs de probabililté dans cette substitution est entre 0 et 1 inclu et leur somme est égale à 1. Nous concluons ainsi que les deux obligations de preuve (evt/assign/pWD1) et (evt/assign/pWD2) sont bien remplies par cet événement. L'invariant de cette machine est aussi toujours respecté par tous les événements.

3.5 Sémantique opérationnelle d'une machine B événementiel probabiliste

Comme indiqué dans la section 2.6.3, la sémantique opérationnelle d'une machine B événementiel standard est exprimée en termes de systèmes de transitions. Dans cette section, nous étendons ce travail et nous établissons que la sémantique opérationnelle d'une machine B événementiel probabiliste est une chaîne de Markov discrète. Contrairement aux travaux de Troubistyna et *al* [91, 93], notre objectif n'est pas de tranformer les machines B événementiel probabilistes vers des chaînes de Markov discrètes pour appliquer des techniques de vérification par *model checking*. Dans notre proposition, nous raisonnons directement sur les machines B événementiel et nous profitons des mécanismes de vérification à base de preuves du B événementiel via l'ajout d'obligations de preuve comme présenté dans la section 3.4. Cette transformation vers une chaîne de Markov n'est faite que pour assurer l'exactitude de notre approche.

3.5.1 Notations

Afin de définir la sémantique d'une machine B événementiel probabiliste, nous introduisons dans cette partie quelques notations de base que nous allons utiliser dans la suite de cette section. Soit $\mathsf{M}=(\bar{v},\mathsf{I}(\bar{v}),\mathsf{PEvts},\mathsf{Init})$ une machine B événementiel et soit σ une valuation donnée des variables de

M. Pour une variable $x \in \bar{v}$ de la machine M, nous notons par $[\sigma]x$ la valeur de x dans la valuation des variables σ. Étant donné une expression $E(\bar{v})$ qui dépend des variables $\bar{v}$ de la machine, nous notons par $[\sigma]E(\bar{v})$ l'évaluation de l'expression $E(\bar{v})$ dans la valuation des variables σ.

Soit e_i un événement probabiliste de la machine M qui possède un ensemble de paramètres $\bar{t}$:

$e_i \ \widehat{=}$
weight
$W_i(\bar{v})$
any
$\bar{t}$
where
$G_i(\bar{t}, \bar{v})$
then
$SP_i(\bar{t}, \bar{v})$
end

Soit σ une valuation des variables de M. Nous notons par θ une valuation des paramètres de e_i. Nous notons par $T_\sigma^{e_i}$ l'ensemble des valuations des paramètres θ telles que la garde de l'événement e_i est satisfaite lorsqu'elle est évaluée dans σ et θ. Formellement, cet ensemble s'écrit $T_\sigma^{e_i} = \{\theta \mid [\sigma, \theta]G_i(\bar{t}, \bar{v}) = true\}$. Nous rappelons que dans notre proposition, les valeurs des paramètres sont choisies d'une manière uniforme à partir de cet ensemble. Nous notons par $\mathbf{P}_{T_\sigma^{e_i}}$ la distribution uniforme discrète sur l'ensemble $T_\sigma^{e_i}$. Nous notons par $\mathsf{Acts}(\sigma)$ l'ensemble des événements qui peuvent être activés dans la valuation des variables σ, c'est-à-dire $T_\sigma^{e_i} \neq \emptyset$.

Soit σ une valuation des variables de M et soit e_i un événement probabiliste de M. L'événement e_i est activé dans la valuation σ si et seulement si

(a) l'expression du poids de l'événement e_i s'évalue à une valeur strictement positive dans la valuation σ, et

(b) soit e_i ne possède pas de paramètres et sa garde est satisfaite dans la valuation des variables σ, soit e_i possède un (des) paramètre(s) et il existe au moins une valuation de ce (ces) paramètre(s) θ telle que la garde de e_i est satisfaite dans la valuation des variables σ et la valeur des paramètres θ, c'est-à-dire que l'ensemble $T_\sigma^{e_i}$ n'est pas vide ($T_\sigma^{e_i} \neq \emptyset$).

Étant donnée un événement probabiliste e_i, nous notons par $\mathsf{Var}(e_i)$ l'ensemble des variables qui sont modifiées par l'action de e_i, c'est-à-dire les variables dans la partie gauche des substitutions de $SP_i(\bar{t}, \bar{v})$. Nous rappelons aussi que dans l'action d'un événement en B événementiel, une variable ne peut être modifiée que par une seule substitution.

Soit e_i un événement probabiliste ($e_i \in \mathsf{PEvts}$), et soit $x \in \mathsf{Var}(e_i)$. Dans le cas où x est modifié par une substitution énumérée probabiliste ($x := E_1(\bar{t},\bar{v})$ @ $p_1 \oplus \ldots \oplus E_m(\bar{t},\bar{v})$ @ p_m) ($m > 1$) , nous notons par $\mathcal{E}_{e_i}(x)$ l'ensemble de toutes les expressions qui peuvent être affectées à la variable x par cette substitution.

$$\mathcal{E}_{e_i}(x) = \{E_1(\bar{t}, \bar{v}), ..., E_m(\bar{t}, \bar{v})\}$$

La probabilité de choisir une expression E_i parmi toutes les expressions E_i ($1 < i < m$) est notée $\mathbf{P}_x^{e_i}(E_i) = p_i$.

Soit $e_i \in \mathsf{PEvts}$ un événement probabiliste, $x \in \mathsf{Var}(e_i)$ une variable modifiée par l'action de e_i, σ, σ' deux valuations des variables $\bar{v}$ et θ une valeur des paramètres de l'événement e_i tel que e_i est

activé dans σ (la garde de e_i est satisfaite dans σ et θ) et tel que l'action de e_i mène le système vers la nouvelle valuation des variables σ'. Si dans l'action de e_i, x est modifiée par une substitution énumérée probabiliste, alors nous notons par $\mathcal{E}_{e_i}(\mathsf{x})|^{\sigma'}_{\sigma,\theta}$ l'ensemble des expressions dans $\mathcal{E}_{e_i}(\mathsf{x})$ telles que l'évaluation de ces expressions dans σ et θ donne la valeur de x dans la valuation σ'.

Formellement,

$$\mathcal{E}_{e_i}(\mathsf{x})|^{\sigma'}_{\sigma,\theta} = \{\mathsf{E} \in \mathcal{E}_{e_i}(\mathsf{x}) \mid [\sigma,\theta](E(\bar{t},\bar{v})) = [\sigma']\mathsf{x}\}$$

Si l'événement e_i n'a pas de paramètre, alors cet ensemble est noté $\mathcal{E}_{e_i}(\mathsf{x})|^{\sigma'}_{\sigma}$.

Si x est modifiée par une substitution probabiliste sous forme de prédicat ($\mathsf{x} :\oplus \mathsf{Q}_\mathsf{x}(\bar{t},\bar{v},\mathsf{x},\mathsf{x}')$), alors nous notons $\mathcal{V}^{e_i}_{\theta,\sigma}(\mathsf{x})$ l'ensemble des valeurs x' qui rendent le prédicat $\mathsf{Q}_\mathsf{x}(\bar{t},\bar{v},\mathsf{x},\mathsf{x}')$ vrai, lorsqu'il est évalué en fonction de σ et θ. Formellement, cet ensemble s'écrit

$$\mathcal{V}^{e_i}_{\theta,\sigma}(\mathsf{x}) = \{\mathsf{x}' \mid [\sigma,\theta]Q_\mathsf{x}(\bar{t},\bar{v},\mathsf{x}') = true\}$$

Si e_i n'a pas de paramètres, alors cet ensemble est noté par $\mathcal{V}^{e_i}_{\sigma}$ (x).

Soit e_i un événement probabiliste et soit x une variable de $\mathsf{Var}(\mathsf{e}_i)$. Étant donnée une valuation initiale des variables σ, une valeur des paramètres θ et une valuation destination des variables σ', nous notons $P^{e_i}_{\sigma,\theta}(\mathsf{x},\sigma')$ la probabilité que la variable x prenne la valeur $[\sigma']\mathsf{x}$ lors de l'exécution de e_i à partir de σ avec la valeur des paramètres θ. Si e_i n'a pas de paramètres, alors cette probabilité est notée $P^{e_i}_{\sigma}(\mathsf{x},\sigma')$. Dans ce qui suit, nous supposons que l'événement e_i possède un ensemble de paramètres. Formellement, cette probabilité est donnée comme suit.

1. Si x est modifiée par une substitution énumérée probabiliste, alors :

$$P^{e_i}_{\sigma,\theta}(\mathsf{x},\sigma') = \sum_{E \in \mathcal{E}_{e_i}(\mathsf{x})|^{\sigma'}_{\sigma,\theta}} P^{e_i}_{\mathsf{x}}(E)$$

2. Si x est modifiée par une substitution probabiliste sous forme de prédicat, alors :

$P^{e_i}_{\sigma,\theta}(\mathsf{x},\sigma') = \frac{1}{card(\mathcal{V}^{e_i}_{\theta,\sigma}(\mathsf{x}))}$ si $[\sigma']\mathsf{x} \in \mathcal{V}^{e_i}_{\theta,\sigma}(\mathsf{x})$ and 0 dans le cas contraire.

3.5.2 Construction du système de transitions probabiliste correspondant à une machine probabiliste

Informellement, la sémantique opérationnelle d'une machine B événementiel probabiliste M=($\bar{v}$, $\overline{\mathsf{ctx}}$, I($\bar{v}$), PEvts, Init) est exprimée sous forme d'un système de transitions probabiliste $[\![M]\!] = (S, s_0, Acts, P)$ où les états, les actions et l'état initial sont obtenus d'une manière similaire à celle d'une machine B événementiel standard. La différence majeure avec la sémantique d'une machine B événementiel standard concerne les transitions. Puisque nous introduisons des probabilités dans B événementiel, les transitions doivent refléter ces probabilités. Nous expliquons dans ce qui suit comment nous obtenons les valeurs des probabilités des transitions. Soit $\mathsf{e}_i \in$ PEvts un événement probabiliste, $\mathsf{x} \in \bar{v}$ une variable de la machine et s, s' deux états de $[\![M]\!]$ tels que (s, e_i, s') est une transition. La probabilité d'une transition (s, e_i, s') correspond au produit de

(1) la probabilité que l'événement e_i est choisi pour être exécuté parmi tous les événements activés dans s,

(2) la probabilité de choisir chaque valeur des paramètres θ,

(3) la probabilité globale que chaque variable modifiée prenne la valeur donnée dans s' avec les valeurs des paramètres θ.

Nous présentons dans ce qui suit la définition formelle de la sémantique d'une machine B événementiel probabiliste.

Définition 6 *(Sémantique opérationnelle d'une machine B événementiel probabiliste)*
La sémantique opérationnelle d'une machine B événementiel probabiliste M=($\bar{v}$,I($\bar{v}$),PEvts, Init) est exprimée en termes de système de transitions probabiliste $[\![M]\!] = (S, s_0, Acts, P)$ où :

- *S est l'ensemble des états du système. Ces états correspondent aux différentes valuations des variables du système.*
- *$s_0 \in S$ est l'état initial du système, il est obtenu après l'exécution de l'événement d'initialisation Init .*
- *$Acts$ correspond à l'ensemble des noms des événements*
- *$P : S \times Acts \times S \to [0, 1]$ est la fonction de probabilité telle que pour un état s, pour une action $e_i \in Acts$ et un état $s' \in S$, nous avons*

  $P(s, e_i, s') = 0$ si $e_i \notin Acts(s)$ ou si $\exists x \in X \backslash Var(e_i)$ tel que $[s]\mathsf{x} \neq [s']\mathsf{x}$.

  Dans le cas contraire :

$$P(s, e_i, s') = \underbrace{\frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)}[s]W_j(\bar{v})}}_{(1)} \times \sum_{\theta \in T_s^{e_i}} \Big(\underbrace{P_{T_s^{e_i}}(\theta)}_{(2)} \times \underbrace{\prod_{\mathsf{x} \in Var(e_i)} P_{s,\theta}^{e_i}(\mathsf{x}, s')}_{(3)} \Big)$$

En fonction de l'événement, cette fonction peut être simplifiée :

1. *Si l'événement e_i n'a aucun paramètre et si son action est purement déterministe alors :*

$$P(s, e_i, s') = \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)}[s]W_j(\bar{v})}$$

2. *Si l'événement e_i possède des paramètres et si son action est purement déterministe :*

$$P(s, e_i, s') = \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)}[s]W_j(\bar{v})} \times \sum_{\theta \in T_s^{e_i}} \Big(P_{T_s^{e_i}}(\theta) \Big)$$

3. *Si l'événement e_i n'a pas de paramètres mais que son action est probabiliste :*

$$P(s, e_i, s') = \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)}[s]W_j(\bar{v})} \times \prod_{\mathsf{x} \in Var(e_i)} P_s^{e_i}(\mathsf{x}, s') \Big)$$

Dans ce qui suit, nous montrons que la sémantique opérationnelle d'une machine B événementiel probabiliste est une chaîne de Markov discrète.

Remarque

Étant donné que nous avons pour but de prouver que la sémantique opérationnelle d'un modèle B événementiel purement probabiliste est une chaîne de Markov discrète, nous définissons directement une relation de transition P sous forme de fonction plutôt que la version plus générale de la définition 4.

3.5.3 Sémantique opérationnelle exprimée en termes de chaîne de Markov discrète

Proposition 1 *La sémantique opérationnelle d'une machine B événementiel probabiliste M qui satisfait les obligations de preuve présentées dans la section 3.4 est une chaîne de Markov discrète.*

Preuve Nous devons prouver que dans chaque état $s \in [\![M]\!]$, la somme des probabilités des transitions sortantes de s est égale à 1.

Soit M=($\bar{v}$, I($\bar{v}$), PEvts, Init) une machine B événementiel probabiliste, $\bar{v} = (\mathsf{x}_1, \mathsf{x}_2, ..., \mathsf{x}_n)$ l'ensemble des variables de M et $s \in S$ un état de $[\![M]\!]$. Nous supposons que chaque variable x_i dans $\bar{v}$ prend une valeur à partir d'un ensemble X_i.

Nous rappelons que la probabilité d'une transition (s, e_i, s') est égale à 0 si $e_i \notin \mathsf{Acts}(s)$ ou $\exists x \in \bar{v} \backslash \{Var(e_i)\} \mid [s]x \neq [s']x$. Dans le cas contraire :

$$P(s, e_i, s') = \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\bar{v})} \times \sum_{\theta \in T_s^{e_i}} \Big(P_{T_s^{e_i}}(\theta) \times \prod_{x \in Var(e_i)} P_{s,\theta}^{e_i}(x, s') \Big)$$

Afin de prouver que pour tout $s \in S$, $P(s, ., .)$ est une distribution de probabilité sur $\mathsf{Acts}(s) \times S$, nous devons montrer que

1) $P(s, e_i, s') \in [0, 1]$ pour chaque (s, e_i, s'), et

2) pour chaque $s \in S$, $\sum_{e_i \in \mathsf{Acts}(s)} \sum_{s' \in S} P(s, e_i, s') = 1$.

Initialement, nous remarquons que (1) est une conséquence directe des obligations de preuve (event/WGHT/NAT), (event/param/pWD), (event/assign/pWD1), (event/assign/pWD2) et (event/assign/pWD3). En effet,

- (event/WGHT/NAT) assure que $0 \leq \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\bar{v})} \leq 1$,
- (event/param/pWD) assure que $0 \leq P_{T_s^{e_i}}(\theta) \leq 1$ pour chaque $\theta \in T_s^{e_i}$ et que $\sum_{\theta \in T_s^{e_i}} P_{T_s^{e_i}}(\theta) = 1$, et
- (event/assign/pWD1), (event/assign/pWD2) et (event/assign/pWD3) assure que $0 \leq \prod_{x \in Var(e_i)} P_{s,\theta}^{e_i}(x, s') \leq 1$ pour chaque $\theta \in T_s^{e_i}$.

En plus, (2) est dérivé comme suit :
Par définition,

$$\sum_{s' \in S, e_i \in \mathsf{Acts}(s)} P(s, e_i, s') = \sum_{e_i \in \mathsf{Acts}(s)} \sum_{s' \in S} \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\bar{v})} \times \sum_{\theta \in T_s^{e_i}} \Big(P_{T_s^{e_i}}(\theta) \times \prod_{x \in Var(e_i)} P_{s,\theta}^{e_i}(x, s') \Big)$$

Puisque seulement $P_{s,\theta}^{e_i}(x, s')$ dépend de s', ceci devient

$$\sum_{s' \in S, e_i \in \mathsf{Acts}(s)} P(s, e_i, s') = \sum_{e_i \in \mathsf{Acts}(s)} \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\bar{v})} \times \sum_{\theta \in T_s^{e_i}} \Big(P_{T_s^{e_i}}(\theta) \times \sum_{s' \in S} \prod_{x \in Var(e_i)} P_{s,\theta}^{e_i}(x, s') \Big)$$

Soit $S_1 = \{s' \in S \mid \forall x \in \bar{v} \backslash Var(e).[s]x = [s']x\}$. Nous remarquons que $P_{s,\theta}^{e_i}(x, s') = 0$ pour chaque $s' \notin S_1$. En conséquence,

$$\sum_{s' \in S, e_i \in \mathsf{Acts}(s)} P(s, e_i, s') = \sum_{e_i \in \mathsf{Acts}(s)} \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\bar{v})} \times \sum_{\theta \in T_s^{e_i}} \Big(P_{T_s^{e_i}}(\theta) \times \underbrace{\sum_{s' \in S_1} \prod_{x \in Var(e_i)} P_{s,\theta}^{e_i}(x, s')}_{(\mathbf{A})} \Big)$$

Maintenant, nous simplifions $(\mathbf{A})$. Pour chaque $x \in Var(e_i)$, rappelons que :

- $P^{e_i}_{s,\theta}(x, s') = \sum_{E \in \mathcal{E}_{e_i}(x)|^{s'}_{s,\theta}} P^{e_i}_x(E)$ si x est modifiée par une substitution probabiliste énumérée, et
- $P^{e_i}_{s,\theta}(x, s') = \frac{1}{card(\mathcal{V}^{e_i}_{\theta,s}(x))}$ si x est modifiée par une substitution probabiliste sous forme de prédicat.

En conséquence, $P^{e_i}_{s,\theta}(x, s')$ ne dépend pas entiérement de s' mais dépend uniquement de la valeur de x dans l'état s' : $v'_x = [s']x \in X$.

Étant donné $x \in \bar{v}$ et $v'_x = [s']x \in X$, nous notons $F^{s,\theta,e_i}_x(v'_x) = P^{e_i}_{s,\theta}(x, s')$ si $x \in Var(e_i)$.

Pour $\bar{v} = \{x_1, \ldots, x_n\}$, nous avons $S_1 = \{(v'_{x_1}, \ldots, v'_{x_n}) | v'_{x_i} \in X_i$ si $x_i \in Var(e_i)$ et $v'_{x_i} = [s]x_i$ dans le cas contraire$\}$.
Supposons que $Var(e_i) = \{x_1, \ldots, x_k\}$ avec $k \leq n$, nous pouvons ainsi réécrire $(\mathbf{A})$ comme suit :

$$
\begin{aligned}
(\mathbf{A}) = \sum_{s' \in S_1} \prod_{x_i \in Var(e_i)} F^{s,\theta,e_i}_x(v'_{x_i}) &= \sum_{v'_{x_1} \in X_1} \sum_{v'_{x_2} \in X_2} \cdots \sum_{v'_{x_k} \in X_k} (\prod_{i=1}^{k} F^{s,\theta,e_i}_x(v'_{x_i})) \\
&= \sum_{v'_{x_1} \in X_1} \sum_{v'_{x_2} \in X_2} \cdots \sum_{v'_{x_k} \in X_k} (F^{s,\theta,e_i}_{x_1}(v'_{x_1}) \cdot F^{s,\theta,e_i}_{x_2}(v'_{x_2}) \cdots F^{s,\theta,e_i}_{x_k}(v'_{x_k})) \\
&= \sum_{v'_{x_1} \in X_1} F^{s,\theta,e_i}_{x_1}(v'_{x_1}) \cdot \sum_{v'_{x_2} \in X_2} F^{s,\theta,e_i}_{x_2}(v'_{x_2}) \cdots \sum_{v'_{x_k} \in X_k} F^{s,\theta,e_i}_{x_k}(v'_{x_k}) \\
&= \prod_{x_i \in Var(e_i)} \sum_{v'_{x_i} \in X_i} F^{s,\theta,e_i}_{x_i}(v'_{x_i})
\end{aligned}
$$

Pour (event/assign/pWD1), (event/assign/pWD2) et (event/assign/pWD3), nous avons $\sum_{v'_{x_i} \in X_i} F^{s,\theta,e_i}_{x_i}(v'_{x_i}) = 1$ pour chaque $x_i \in Var(e_i)$, ainsi $(\mathbf{A}) = \prod_{x_i \in Var(e_i)} [\sum_{v'_{x_i} \in X_i} F^{s,\theta,e_i}_{x_i}(v'_{x_i})] = 1$

En conséquence,

$$
\sum_{s' \in S, e_i \in \mathsf{Acts}(s)} P(s, e_i, s') = \sum_{e_i \in \mathsf{Acts}(s)} \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\bar{v})} \times \sum_{\theta \in T^{e_i}_s} (P_{T^{e_i}_s}(\theta))
$$

De plus, par construction, nous avons $\sum_{\theta \in T^{e_i}_s} P_{T^{e_i}_s}(\theta) = 1$. Ainsi,

$$
\sum_{s' \in S, e_i \in \mathsf{Acts}(s)} P(s, e_i, s') = \sum_{e_i \in \mathsf{Acts}(s)} \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\bar{v})} = 1
$$

En conclusion, $P(s, ., .)$ est en effet une distribution de probabilité sur $\mathsf{Acts}(s) \times S$ pour tout $s \in S$ et ainsi $[\![M]\!]$ est une DTMC. □

3.5.4 Étude de cas : sémantique opérationnelle de la machine P2P$_p$ du protocole pair à pair

Afin d'illustrer les nouveaux choix probabilistes introduits à la place des choix non-déterministes et de montrer d'une manière détaillée et concrète le calcul des probabilités des transitions de la sémantique d'une machine B événementiel probabiliste, nous donnons dans la figure 3.11 les étapes initiales de la construction de la chaîne de Markov correspondante à la machine P2P$_p$. Afin

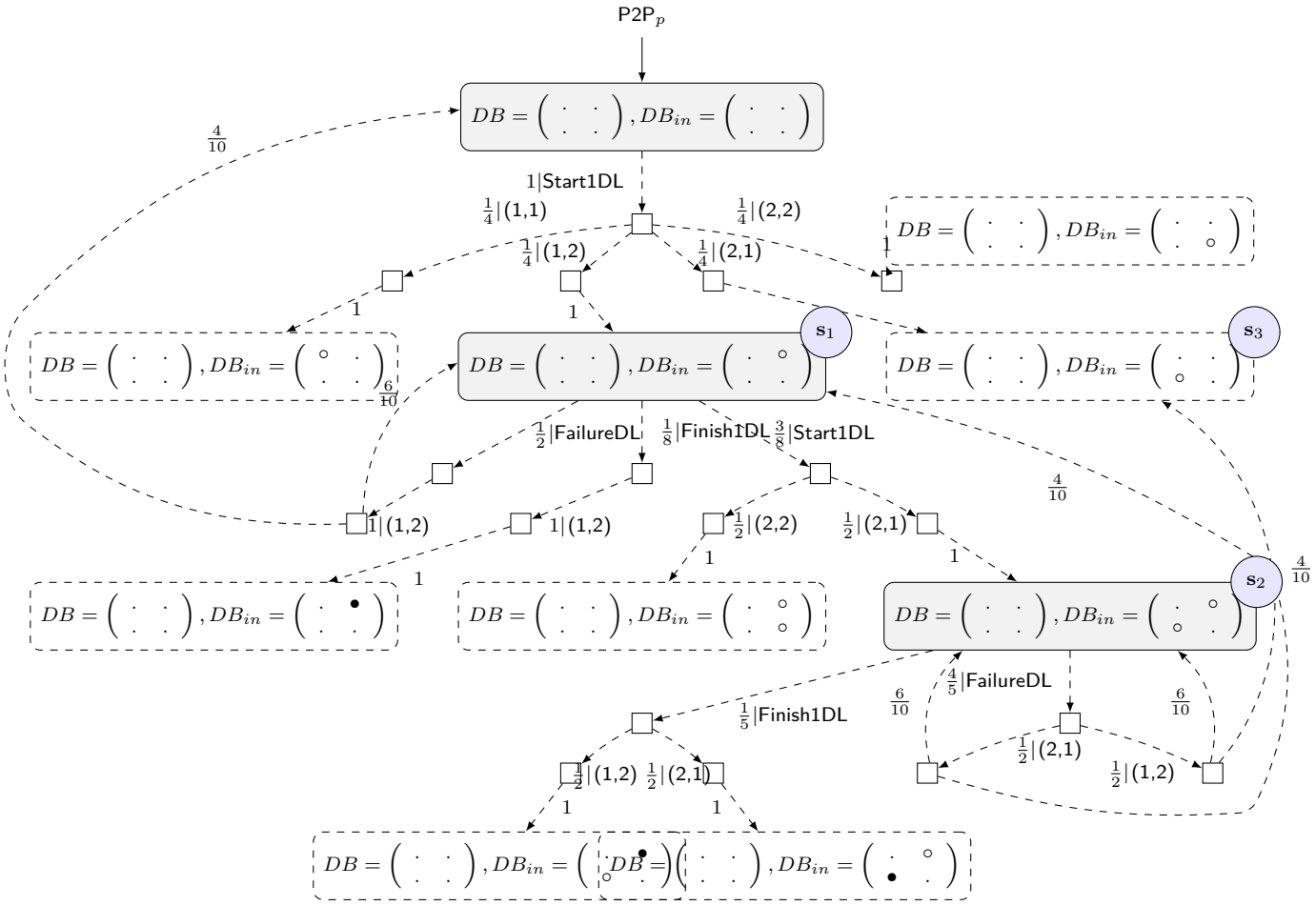

FIGURE 3.11 – Extrait de la construction détaillée de la chaîne de Markov discrète du protocole P2P avec N=2 and K=2

de simplifier la construction de cette sémantique, nous fixons le nombre de clients N, ainsi que le nombre de blocs K à 2.

Nous détaillons pour certains cas typiques le calcul des probabilités des transitions, les autres valeurs de probabilités sont facilement calculables. À partir de l'état s_1, trois événements sont activables, FailureDL, Start1DL et Finish1DL.

- Le poids de FailureDL est $\mathsf{N} \times \mathsf{K} - \mathsf{card}(\{\ \mathsf{n} \mapsto \mathsf{b} \mid \mathsf{n} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\ \mathsf{n} \mapsto \mathsf{b}\ ) = \mathsf{finished}\ \})$. Dans l'état s_1, aucun client n'a encore terminé le téléchargement d'aucun bloc, donc ce poids prend la valeur 4. La probabilité d'exécuter cet événement dans cet état correspond au ratio de son poids divisée par la somme des poids des autres événements activés dans s_1. Cette probabilité est donc $\frac{4}{4+3+1} = \frac{1}{2}$.

- Le poids de Start1DL est $\mathsf{N} \times \mathsf{K} - \mathsf{card}(\{\ \mathsf{n} \mapsto \mathsf{b} \mid \mathsf{n} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\ \mathsf{n} \mapsto \mathsf{b}\ ) = \mathsf{incoming}\})$. Dans l'état s_1, il existe un client qui a commencé le téléchargement d'un bloc, donc ce poids prend la valeur 3. La probabilité d'exécuter cet événement dans cet état correspond au ratio de son poids divisée par la somme des poids des événements activés dans s_1, cette probabilité est donc $\frac{3}{4+3+1} = \frac{3}{8}$.

- Le poids de Finish1DL est $\mathsf{card}(\{\mathsf{n} \mapsto \mathsf{b} \mid \mathsf{n} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\ \mathsf{n} \mapsto \mathsf{b}\ ) = \mathsf{finished}\ \}) + 1$. Dans l'état s_1, aucun client n'a encore fini le téléchargement d'un bloc, donc ce poids prend la valeur 1. La probabilité d'exécuter cet événement dans cet état correspond au ratio de

son poids divisée par la somme des poids des autres événements activés dans (s_1), cette probabilité est donc $\frac{1}{4+3+1} = \frac{1}{8}$.

L'événement Finish1DL possède deux paramètres c et b, ces deux paramètres représentent respectivement un client et un bloc et ils sont choisis de telle sorte que le client c a commencé le téléchargement du bloc b. Dans l'état s_1, lorsque l'événement Finish1DL est choisi pour être exécuté, une seule valeur des paramètres pour cet événement est possible. La probabilité de prendre cette valeur est ainsi 1. L'action de Finish1DL est déterministe, sa probabilité est donc 1. Ainsi, la probabilité de la transition sortante de s_1 par l'événement Finish1DL est alors $\frac{1}{8} \times 1 \times 1 = \frac{1}{8}$.

Si l'événement Start1DL est choisi pour être exécuté, alors deux valeurs des paramètres sont possibles. Etant donnée que nous considèrons un choix uniforme sur les valeurs possibles des paramètres, alors chacune des ces valeurs est choisie avec une probabilité $\frac{1}{2}$. L'action de l'événement Start1DL est déterministe, donc sa probabilité d'exécution est 1. La valeur des paramètres $(2, 1)$ (client numéro 2 et bloc numéro 1) permet au système d'aller à l'état s_2 par l'événement Start1DL. Ainsi, la probabilité d'aller vers l'état s_2 à partir de l'état s_1 par l'intermédiaire de l'événement Start1DL est donc $\frac{3}{8} \times \frac{1}{2} \times 1 = \frac{3}{16}$.

Si nous choisisons l'événement FailureDL, une unique valeur des paramètres est possible. L'action de cet événement est probabiliste, menant ainsi le système à deux états (rester dans l'état s_1 avec une probabilité $\frac{6}{10}$ ou aller dans un autre état avec une probabilité $\frac{4}{10}$. La probabilité de boucler sur l'état s_1 par l'intermédiaire de l'événement FailureDL est donc $\frac{1}{2} \times 1 \times \frac{6}{10}$.

Pour l'état s_2, nous présentons ici une seule transition importante. Dans cet état, deux événements peuvent être activés : Finish1DL et FailureDL. Le poids de Finish1DL dans cet état s'évalue à 1 alors que le poids de FailureDL s'évalue à 4 La probabilité d'exécuter FailureDL est ainsi $\frac{4}{5} = \frac{4}{1+4}$. Les paramètres de cet événement peuvent prendre deux valeurs : $(1, 2)$ pour client numéro 1 et bloc numéro 2 et $(2, 1)$ pour client numéro 2 et bloc numéro 1. Chacune de ces valeurs est choisie avec une probabilité $\frac{1}{2}$. L'action de cet événement est probabiliste, menant le système à différents états. L'intérêt de cette transition réside dans le fait que pour différentes valeurs des paramètres, l'action de l'événement emmène le système au même état :

- La probabilité de boucler sur l'état (s_1) par l'intermédiaire de l'événement FailureDL est $\frac{4}{5} \times \frac{1}{2} \times \frac{4}{10} = \frac{4}{25}$
- La probabilité d'aller vers l'état (s_3) par l'intermédiaire de l'événement FailureDL est $\frac{4}{5} \times \frac{1}{2} \times \frac{4}{10} = \frac{4}{25}$
- La probabilité de boucler sur l'état (s_2) est $\frac{4}{5} \times (\underbrace{\frac{1}{2} \times \frac{6}{10}}_{(2,1)} + \underbrace{\frac{1}{2} \times \frac{6}{10}}_{(1,2)}) = \frac{12}{25}$, où (2,1) et (1,2) sont les valeurs des paramètres menant le système à cet état.

Nous présentons un extrait de la chaîne de Markov résultante dans la figure 3.12.

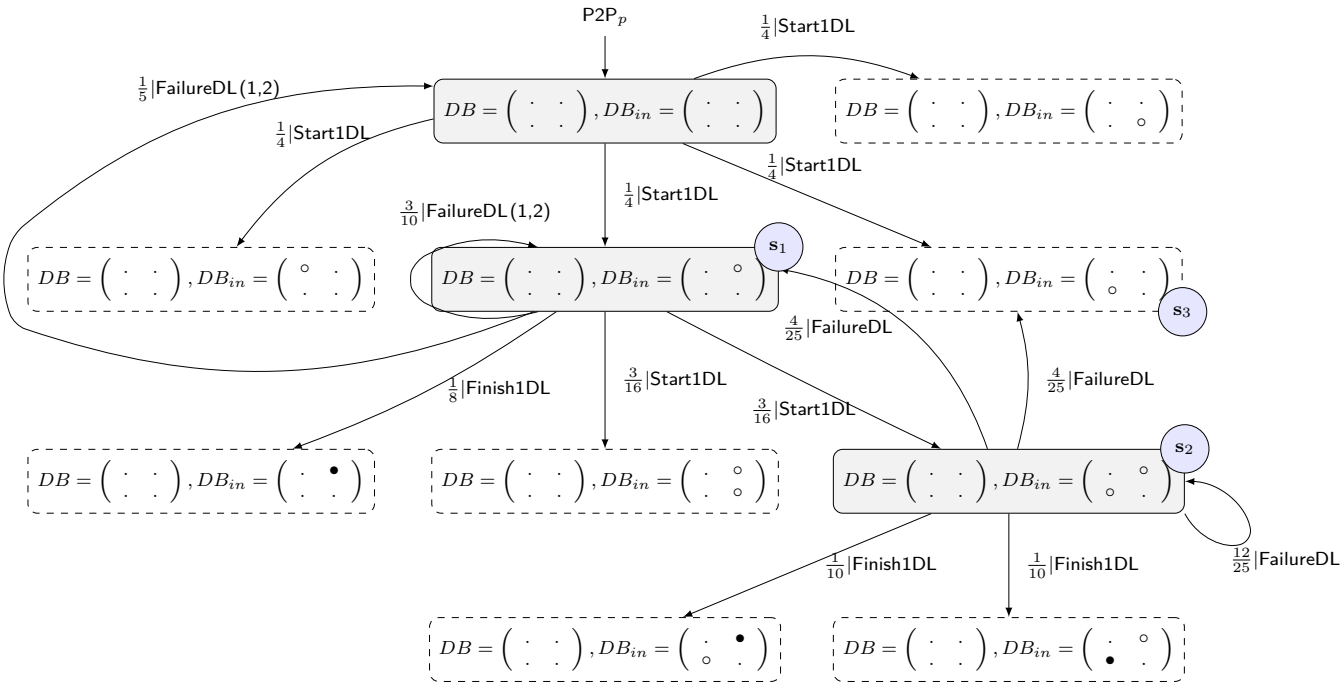

FIGURE 3.12 – Extrait de la chaîne de Markov du protocole P2P, avec N=2 et K=2

4

B Événementiel mixte

4.1 Introduction

Nous rappelons que l'objectif de notre travail consiste à intégrer du raisonnement probabiliste en B événementiel. Dans le chapitre précédent, nous avons présenté une extension probabiliste au B événementiel dans laquelle nous avons remplacé toutes les sources de non-déterminisme par des probabilités. Dans ce chapitre, nous étendons le travail précédent en présentant des modèles B événementiel mixtes, c'est-à-dire, des modèles contenant à la fois du non-déterminisme et des probabilités. Comme mentionné dans le chapitre 1, plusieurs travaux de recherche [78, 58, 98, 53, 91, 92, 93] ont déjà traité l'extension probabiliste du B événementiel conduisant ainsi à des modèles mixtes.

La différence de ces modèles mixtes avec nos modèles mixtes réside dans le lieu d'apparition du non-déterminisme. Dans ces autres travaux, le non-déterminisme au niveau des substitutions est complètement remplacé par des probabilités. Cependant, il reste du non-déterminisme entre les événements activables dans les mêmes valuations des variables et dans le choix des valeurs des paramètres d'un événement. Dans nos modèles mixtes, il y aura à la fois des événements non-déterministes (standard) et des événements probabilistes, le non-déterminisme sera gardé dans les substitutions non-déterministes des événements standard, dans le choix des valeurs des paramètres de ces événements et aussi entre les événements non-déterministes et probabilistes activables dans les même valuations des variables.

Dans ce chapitre, nous présentons les machines B événementiel mixtes dans la section 4.2. Dans la section 4.3, nous étudions la cohérence d'une machine B événementiel mixte. Plus précisément, nous étudions les modifications à apporter aux obligations de preuve standard et aux obligations de preuve spécifique au B événementiel probabiliste afin qu'elles puissent être appliquées aux machines B événementiel mixtes. Nous présentons dans la section 4.4 la sémantique d'une machine B événementiel mixte en termes d'un automate probabiliste. Finalement, nous présentons dans la section 4.5 la description de l'étude de cas du protocole pair à pair en B événementiel mixte.

4.2 Machine B événementiel mixte

4.2.1 Description

Une machine B événementiel mixte est une machine qui contient à la fois du non-déterminisme et des probabilités. Plus précisément, elle contient des événements probabilistes et des événements standard non-déterministes. Les événements standard interagissent avec les événements probabilistes, c'est-à-dire, qu'ils peuvent être activables dans un même état du système.

Comme en B événementiel, une machine B mixte permet la description du comportement dynamique d'un système. La structure d'une machine B événementiel mixte est présentée dans la figure 4.1, elle possède la même structure qu'une machine B événementiel standard ou probabiliste. La seule différence concerne l'ensemble des événements MEvts. Dans une machine mixte, cet ensemble contiendra des événements standards ainsi que des événements probabilistes. Pour une utilisation plus simple des machines mixtes, nous dénoterons dans la suite de ce chapitre une machine mixte par un tuple mch=($\bar{v}$,$\overline{\text{ctx}}$,I($\bar{v}$),Init,MEvts,V($\bar{v}$)). Comme c'est le cas pour les machines B événementiel standard, nous précisons que l'événement d'initialisation d'une machine B événementiel mixte est un événement déterministe. L'ensemble des événements MEvts contient des événements standard et des événements probabilistes, les deux formes de ces événements sont rappelées dans la figure 4.2.

MACHINE
mch
SEES
$\overline{\text{ctx}}$
VARIABLES
$\bar{v}$
INVARIANT
I($\bar{v}$)
VARIANT
V($\bar{v}$)
EVENTS
MEvts
END

FIGURE 4.1 – Structure d'une machine mixte

$e_i \mathrel{\widehat{=}}$
any
$\bar{t}$
where
$G_i(\bar{t},\bar{v})$
then
$S_i(\bar{t},\bar{v})$
end

.

$e_i \mathrel{\widehat{=}}$
weight
$W_i(\bar{v})$
any
$\bar{t}$
where
$G_i(\bar{t},\bar{v})$
then
$SP_i(\bar{t},\bar{v})$
end

FIGURE 4.2 – Formes des événements en B événementiel mixte

Dans une machine B événementiel mixte, il est possible que les événements non-déterministes

n'interagissent pas du tout avec les événements probabilistes, c'est-à-dire que l'ensemble des valuations des variables où certains événements non-déterministes sont activables est disjoint de l'ensemble des valuations où des événements probabilistes sont activables. Nous dénoterons ce type de machines dans la suite par *machine mixte partielle*. Une machine B mixte qui ne contiendrait que des événements probabilistes serait une machine B événementiel purement probabiliste comme décrite dans le chapitre 3.

4.3 Cohérence d'une machine B événementiel mixte

Comme dans le cas du B événementiel non-déterministe et le cas du B événementiel probabiliste, la cohérence d'une machine mixte est définie par des obligations de preuve. Une machine B événementiel mixte est cohérente si elle satisfait ces obligations de preuve.

4.3.1 Cas général

Démontrer qu'une machine mixte est cohérente revient à démontrer que chaque événement de la machine satisfait toutes les obligations de preuve spécifiques à son type. Si c'est une machine non-déterministe, elle doit satisfaire les obligations de preuve standard. Si c'est une machine probabiliste, en plus des obligations de preuve standard adaptées, elle doit satisfaire les obligations de preuve probabilistes. Vu les interactions possibles entre événements non-déterministes et événements probabilistes au sein d'une machine mixte, nous étudions les adaptations nécessaires aux obligations de preuve standards et probabilistes contenant des interactions entre les deux types d'événements.

Soit mch=($\bar{v}$,$\overline{\mathsf{ctx}}$,I($\bar{v}$),Init,MEvts,V($\bar{v}$)) une machine B événementiel mixte, l'ensemble des événements MEvts $= \{\mathsf{e}_1 \ldots \mathsf{e}_i \ldots \mathsf{e}_n\}$ est partitionné en deux sous ensembles MEvts_{nd} et MEvts_p. $\mathsf{MEvts}_{nd} = \{\mathsf{e}_1 \ldots \mathsf{e}_i\}$ est le sous-ensemble d'événements standard non-déterministes alors que $\mathsf{MEvts}_p = \{\mathsf{e}_{i+1} \ldots \mathsf{e}_n\}$ est le sous-ensemble d'événements probabilistes. Pour que la machine M soit cohérente, il faut que :

- Les événements non-déterministes de l'ensemble MEvts_{nd} satisfassent les obligations de preuve standard (event/FIS), (event/INV) présentées dans la section 2.3,
- Les événements probabilistes satisfassent
  - les obligations de preuve standard adaptées au contexte probabiliste (event/pFIS), (event/pINV) présentées dans la section 3.4 et,
  - les obligations de preuve probabilistes (event/WGHT/NAT), (event/param/pWD), (event/assign/pWD1), (event/assign/pWD2) et (event/assign/pWD3) présentées dans la section 3.4.

La seule obligation de preuve en B événementiel qui contient de l'interaction entre événements probabilistes et événements non-déterministes est l'obligation de preuve de l'absence de blocage. Nous présentons son adaptation dans ce qui suit. En B événementiel, cette obligation de preuve consiste à prouver que dans toutes les valuations possibles des variables qui satisfont l'invariant, au moins un événement est activable. En B événementiel standard, un événement est activable dans une valuation donnée des variables si cette valuation des variables satisfait sa garde. En B événementiel probabiliste, l'activation d'un événement probabiliste dans une valuation des variables particulière est différente de celle d'un événement standard. En effet, en plus de sa garde qui doit être satisfaite dans cette valuation, l'évaluation de son poids dans cette valuation doit retourner un entier positif. Dans une machine B mixte, il est possible d'avoir des valuations des variables où des événements standard et des événements probabilistes sont activables en même temps, cette

obligation de preuve va donc consister à assurer qu'il existe au moins un événement standard *ou* un événement probabiliste activable.

Pour un ensemble d'événements $\mathsf{MEvts} = \{\mathsf{e}_1 \ldots \mathsf{e}_i \ldots \mathsf{e}_n\}$ partitionné en un sous-ensemble d'événements non-déterministes $\mathsf{MEvts}_{nd} = \{\mathsf{e}_1 \ldots \mathsf{e}_i\}$ et un sous-ensemble d'événements probabilistes $\mathsf{MEvts}_p = \{\mathsf{e}_{i+1} \ldots \mathsf{e}_n\}$, cette obligation de preuve est donnée formellement par :

machine/mDLF

$$\mathsf{I}(\bar{v})$$
$$\vdash$$
$$(\mathsf{G}_1(\bar{t},\bar{v}) \vee \ldots \vee \mathsf{G}_m(\bar{t},\bar{v})) \vee ((\mathsf{G}_{m+1}(\bar{t},\bar{v}) \wedge \mathsf{W}_{m+1}(\bar{v}) > 0) \vee \ldots \vee (\mathsf{G}_n(\bar{t},\bar{v}) \wedge \mathsf{W}_n(\bar{v}) > 0))$$

4.3.2 Cas des machines mixtes partielles

Comme mentionné précédemment, il est possible d'avoir des machines B mixtes dont les valuations des variables où des événements non-déterministes sont activables sont séparés des valuations des variables où des événements probabilistes sont activables, c'est les machines mixtes *partielles*. Nous formalisons cela par une nouvelle obligation de preuve permettant d'assurer que nous avons bien à faire à une machine mixte.

Activation d'événements standards :

Cette obligation de preuve assure que, dans chaque valuation des variables de la machine où au moins un événement non-déterministe de MEvts_{nd} est activable, aucun événement probabiliste de MEvts_p n'est activable. Cette obligation de preuve est formellement présentée comme suit :

mixedEB/csrt1

$$\mathsf{I}(\bar{v}) \wedge$$
$$\mathsf{G}_1(\bar{t},\bar{v}) \vee \ldots \vee \mathsf{G}_i(\bar{t},\bar{v})$$
$$\vdash$$
$$\neg((\mathsf{G}_{i+1}(\bar{t},\bar{v}) \wedge \mathsf{W}_{i+1}(\bar{v}) > 0) \vee \ldots \vee (\mathsf{G}_n(\bar{t},\bar{v}) \wedge \mathsf{W}_n(\bar{v}) > 0))$$

Note 3 *La contrainte symétrique sur les valuations des variables où des événements probabilistes sont activables est déduite automatiquement à partir de cette obligation de preuve.*

Une machine B événementiel qui satisfait cette obligation de preuve est une *machine mixte partielle*. Dans une machine mixte partielle, il n'y a pas d'interaction entre événements non-déterministes et événements probabilistes, l'obligation de preuve concernant l'absence de blocage est donnée par :

machine/DLF
$\mathsf{I}(\bar{v})$
$\vdash$
$(\mathsf{G}_1(\bar{t},\bar{v}) \vee \ldots \vee \mathsf{G}_m(\bar{t},\bar{v}))$

machine/pDLF
$\mathsf{I}(\bar{v})$
$\vdash$
$(\mathsf{G}_{m+1}(\bar{t},\bar{v}) \wedge \mathsf{W}_{m+1}(\bar{v}) > 0) \vee \ldots \vee ((\mathsf{G}_n(\bar{t},\bar{v}) \wedge \mathsf{W}_n(\bar{v}) > 0))$

4.4 Sémantique opérationnelle d'une machine B événementiel mixte

Dans cette section, nous étudions la sémantique opérationnelle d'une machine B événementiel mixte. Comme présenté dans les sections 2.6.3 et 3.5, la sémantique opérationnelle d'une machine B événementiel standard est exprimée en termes de systèmes de transitions alors que celle d'une machine B événementiel probabiliste est exprimée en termes d'une chaîne de Markov discrète. Nous étendons ces travaux et nous présentons la sémantique opérationnelle d'une machine B événementiel mixte en termes d'un automate probabiliste. Comme indiqué dans la section 3.5, nous précisons que notre objectif ne consiste pas à transformer les machines B événementiel mixtes vers des automates probabilistes et appliquer des techniques de vérification de modèles pour les vérifier. L'objectif ici consiste à valider l'exactitude de notre approche.

4.4.1 Construction de la sémantique opérationnelle d'une machine B événementiel mixte

Nous souhaitons définir la sémantique opérationnelle d'une machine B événementiel mixte en termes d'un automate probabiliste. Informellement, la sémantique opérationnelle d'une machine B événementiel mixte $\mathsf{M}=(\bar{v},\overline{\mathsf{ctx}},\mathsf{I}(\bar{v}),\mathsf{Init},\mathsf{MEvts},\mathsf{V}(\bar{v}))$ est exprimée en termes d'un système de transitions probabiliste $[\![M]\!]=(S, Acts, T, s_0)$ dont les états sont obtenus de la même manière que dans le cas des machines B standard ou des machines B purement probabilistes. Dans la sémantique de ces machines, la différence majeure avec la sémantique d'une machine B standard et la sémantique d'une machine B purement probabiliste concerne les transitions. Nous rappelons que dans la sémantique d'une machine B standard, les transitions ne sont pas équipées des probabilités et le choix entre les transitions activables dans un état donné se fait d'une manière non-déterministe. Dans la sémantique d'une machine B purement probabiliste exprimée en termes de chaînes de Markov discrètes, les transitions sont équipées des valeurs de probabilités et le choix entre plusieurs transitions activables dans un état donné est fait d'une manière probabiliste.

Pour l'état initial s_0, nous rappelons que nous considérons des machines B événementiel mixtes avec un événement d'initialisation déterministe. En conséquence, nous obtenons un seul état initial s_0, les valuations des variables dans cet état correspondent aux valuations obtenues après l'exécution de l'événement d'initialisation.

Dans ce qui suit, nous définissons la sémantique d'une machine mixte. Principalement, nous expliquons comment nous construisons les transitions. Rappelons qu'une machine mixte contient des

événements non-déterministes et des événements probabilistes. En général, les gardes des événements probabilistes s'intersectent avec les gardes des événements standards. Ainsi, il est possible d'avoir des événements standard et des événements probabilistes qui sont activables en même temps. Puisqu'une machine mixte contient des événements standard et probabilistes, alors la sémantique correspondante contiendra à la fois des choix non-déterministes et des choix probabilistes, plus précisément, un choix non-déterministe entre toutes les transitions activables suivi d'un choix probabiliste (si nécessaire). Pour qu'on puisse définir la sémantique en termes d'un automate probabiliste, nous introduisons deux types de transitions probabilistes, nous les présentons dans ce qui suit.

- Pour les événements standard, et pour chaque valuation des variables obtenues après l'exécution de l'action d'un événement standard, nous avons une transition correspondante qui est une transition probabiliste annotée avec la valeur de probabilité 1. Ainsi, pour chaque événement standard, le nombre de transitions correspondantes correspond au nombre de valuations des variables possiblement obtenues après l'exécution de l'action de l'événement.

- Pour les événements probabilistes, et pour chaque valuation des variables obtenue après l'exécution de l'action d'un événement probabiliste, nous avons une transition probabiliste correspondante annotée avec une valeur de probabilité calculée de la même manière que les transitions probabilistes présentées dans la section 3.5 du chapitre 3. Étant donnée un état s et un événement e_i activable dans l'état s et qui mène le système vers un état s', nous rappelons que la probabilité de la transition de s vers s' par l'intermédiaire de e_i est donnée par :

$$P(s, e_i, s') = \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\bar{v})} \times \sum_{\theta \in T_s^{e_i}} \left(P_{T_s^{e_i}}(\theta) \times \prod_{\mathsf{x} \in Var(e_i)} P_{s,\theta}^{e_i}(\mathsf{x}, s') \right)$$

  Toutes les transitions correpondantes à des événements probabilistes sont regroupées sous la même étiquette $Prob$. La somme des transitions étiquetées par $Prob$ fera bien 1. Ainsi, $Prob$ définira bien une distribution de probabilité correcte.

Dans un état donné, il y aura un choix non-déterministe entre toutes les transitions probabilistes correspondantes à des événements standards et l'ensemble des transitions étiquetées par $Prob$. Il y aura ensuite un choix probabiliste entre toutes les transitions étiquetées par $Prob$. Concrétement, ceci correspond à un choix non-déterministe entre plusieurs distributions de probabilités. Afin de mieux expliquer la construction des transitions, nous donnons deux exemples de machines B événementiel mixtes et nous illustrons leur sémantique.

4.4.2 Exemples d'illustration

Exemple1 : cas d'une machine mixte partielle Considérons la machine mixte partielle Exemple1 donnée dans la figure 4.3. Cette machine contient une seule variable x de type entier, x est initialement initialisée à 0. Elle contient quatre événements : e1, e2, e3 et e4. Les événements e1 et e2 sont deux événements standard, ils sont activables lorsque la variable x est égale à 0. Les événements e3 et e4 sont deux événements probabilistes, ils sont activables lorsque la variable x est égale à 2.

Nous détaillons dans la figure 4.4 la sémantique de Exemple1.

- Dans l'état (x=0), deux événements sont activables :
  - L'événement e1 mène le système vers l'état (x=1) par une transition annotée avec une probabilité de 1,
  - L'événement e2 possède une substitution non-déterministe x :∈ {2,3}, il mène le système vers deux états possibles :

MACHINE
Exemple1
VARIABLES
x
INVARIANTS
$x \in NAT$
EVENTS
Init $\widehat{=}$
$x := 0$
end
e1 $\widehat{=}$
when $x = 0$ **then**
$x := 1$
end
e2 $\widehat{=}$
when $x = 0$ **then**
$x :\in \{2,3\}$
end

e3 $\widehat{=}$
weight
1
when $x = 2$ **then**
$x := 4$
end

e4 $\widehat{=}$
weight
2
when $x = 2$ **then**
$x := 4@3/4 \oplus 5@1/4$
end

END

.

FIGURE 4.3 – Une machine B événementiel mixte partielle

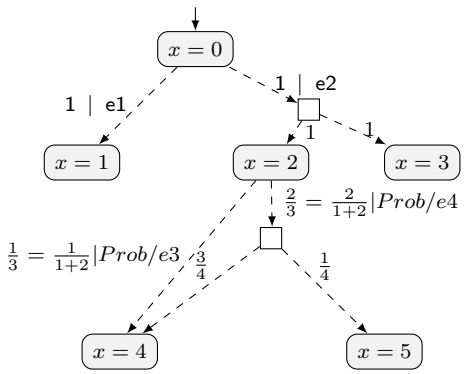

FIGURE 4.4 – Construction détaillée de la sémantique de la machine Exemple1

- si la valeur 2 est choisie pour etre affectée à la variable x, alors la transition correpondante mène le système vers l'état (x=2) et est annotée par la valeur de probabilité 1,
- si la valeur 3 est choisie pour etre affectée à la variable x, alors la transition correpondante mène le système vers l'état (x=3) et est annotée par la valeur de probabilité 1,

Dans l'état (x=2), deux événements sont activables : e3 et e4. Nous expliquons ci-dessous la construction des transitions correpondantes.

– L'événement e3 a un poids de 1. La probabilité d'exécuter cet événement dans cet état est $\frac{1}{3} = \frac{1}{1+2}$, elle correspond au ratio du poids de $e3$ divisé par la somme des poids de $e3$ et $e4$. Étant donné que son action est déterministe (x:=4), la transition correspondante mène le système vers l'état avec une valeur de probabilité de $\frac{1}{3}$. Puisque plusieurs événements probabilistes sont activables dans (x=2), la transition correspondante sera

étiquetée par $Prob/e3$.

– L'événement e4 a un poids de 2. La probabilité d'exécuter cet événement dans cet état est $\frac{2}{1+2} = \frac{2}{3}$, elle correspond au ratio du poids de e4 divisé par la somme des poids de $e3$ et $e4$. L'action de cet événement est non-déterministe, il ya donc deux transitions étiquetées par $Prob/e4$.
  * une qui correspond à l'affectation de la valeur 4 à la variable x et qui est annotée par la valeur de probabilité de $\frac{3}{4}$.
  * une qui correspond à l'affectation de la valeur 5 à la variable x et qui est annotée par la valeur de probabilité de $\frac{1}{4}$.

La figure 4.5 donne la sémantique de la machine Exemple1 sous forme d'un automate probabiliste.

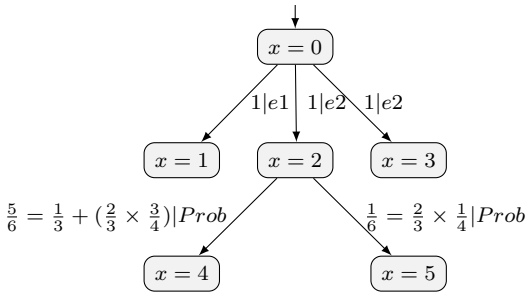

FIGURE 4.5 – Sémantique de la machine Exemple1

Dans l'état (x=0), trois transitions sont activables. La première transition est étiquetée par e1 et annotée par une probabilité de 1. Les deux autres transitions sont étiquetées par e2, et chacune est annotée par une valeur de probabilité de 1. Le choix entre ces transitions est non-déterministe. Dans l'état (x=2), deux transitions étiquetées par $Prob$ sont activables. La première annotée par la valeur de probabilité $\frac{5}{6}$ mène le système vers l'état (x=4), elle correspond au regroupement des transitions de l'état (x=2) vers l'état (x=4) correpondante à l'événement e3 et de (x=2) vers l'état (x=4) correpondante à l'événement e4 ($\frac{5}{6} = \frac{1}{3} + (\frac{2}{3} \times \frac{3}{4})$). La deuxième annotée par la valeur de probabilité $\frac{1}{6}$ mène le système vers l'état (x=5) ($\frac{1}{6} = \frac{2}{3} \times \frac{1}{4}$). La somme des transitions étiquetées par $Prob$ est bien égale à 1.

Exemple2 : cas général Considérons la machine B événementiel mixte Exemple2 donnée dans la figure 4.6. Cette machine contient une seule variable x qui est initialisée à 0. Elle contient quatre événements, deux non-déterministes e1 et e2 et deux probabilistes e3 et e4. Tous les événements sont activables dans l'état (x=0). La sémantique correspondante est donnée dans la figure 4.7. Comme précédemment, $Prob$ regroupe l'ensemble des transitions probabilistes correpondantes aux événements e3 et e4. Nous remarquons que la somme des probabilités des transitions étiquetées par $Prob$ est bien égale à 1. Les autres transitions sont annotées par la valeur de probabilité de 1. Dans l'état (x=0), le choix entre ces transitions non-déterministes et l'ensemble de transitions étiquetées par $Prob$ est un choix non-déterministe.

4.4.3 Sémantique opérationnelle

Nous présentons dans ce qui suit la définition formelle de la sémantique d'une machine B événementiel mixte en termes de système de transitions probabiliste et nous montrons qu'elle correspond bien à un automate probabiliste.

MACHINE
Exemple2
VARIABLES
x
INVARIANTS
$x \in NAT$
EVENTS
Init $\widehat{=}$
$x := 0$
end

e1 $\widehat{=}$
when $x = 0$ **then**
$x := 1$
end

e2 $\widehat{=}$
when $x = 0$ **then**
$x :\in \{2,3\}$
end

e3 $\widehat{=}$
weight
1
when $x = 0$ **then**
$x := 4$
end

e4 $\widehat{=}$
weight
2
when $x = 0$ **then**
$x := 4@3/4 \oplus 3@1/4$
end

END

.

FIGURE 4.6 – Une machine B événementiel mixte

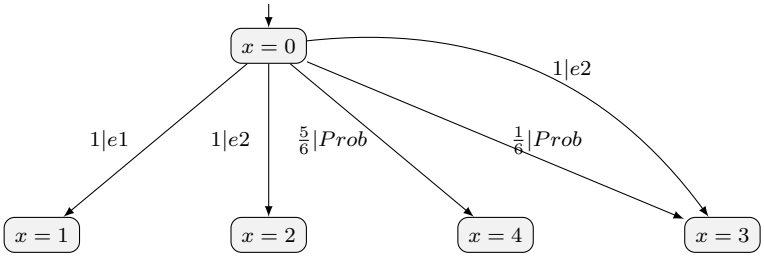

FIGURE 4.7 – Sémantique de la machine Exemple2

Définition 7 *(Sémantique opérationnelle d'une machine B événementiel mixte)*
La sémantique opérationnelle d'une machine B événementiel mixte M=($\bar{v}$,$\overline{\text{ctx}}$,I($\bar{v}$),Init,MEvts,V($\bar{v}$)) est exprimée en termes d'un système de transitions probabiliste $[\![M]\!] = (S, s_0, \text{MActs}, T)$ où :

- *S est l'ensemble des états du système, ces états correspondent aux différentes valuations des variables du système.*
- *s_0 est l'état initial obtenu après l'exécution de l'événement de l'initialisation,*
- MActs *correspond à l'ensemble des noms des événements non-déterministes ainsi que l'action $Prob$ (*MActs $= Acts_{nd} \cup \{Prob\}$*),*
- *$T \subseteq S \times$* MActs *$\times (S \times [0,1])$ est telle que*
  - *$(s, e, \delta) \in T$ si $s \in S$, $e \in Acts_{nd}$ et $\exists s' \in [s]S_e$ tel que $\delta(s') = 1$ et $\delta(s'') = 0 \; \forall \; s'' \neq s'$, ou,*
  - *$(s, Prob, \delta) \in T$ si $\forall \; s' \in [s]S_e$, $\delta(s') = \sum_{e \in Acts_p} P(s, e, s')$ (P est définie dans la section 3.5).*

Dans la proposition suivante, nous montrons que comme prévu, la sémantique opérationnelle d'une machine B événementiel mixte est un automate probabiliste.

Proposition 2 *La sémantique opérationnelle d'une machine B événementiel mixte qui satisfait les obligations de preuve présentée dans la section 4.3 est exprimée en termes d'un automate probabiliste.*

Preuve Nous devons montrer que pour tous les états $s \in S$ et $(s, e, \delta) \in T$, δ est une distribution correcte sur S.

- Si $e \in \mathsf{Acts}_{nd}$, nous déduisons automatiquement que δ est une distribution correcte à partir de la définition de δ.
- Dans le cas contraire, nous avons $e = Prob$ et par la Proposition 1, nous avons

$$\sum_{s' \in S} \delta(s') = \sum_{s' \in S} \sum_{e_k \in \mathsf{Acts}_p} P(s, e_k, s') = 1$$

4.5 Etude de cas : protocole pair à pair

Modèle mixte Afin d'illustrer notre proposition de B événementiel mixte, commençons par reprendre le cas d'étude du protocole pair à pair introduit dans la section 2.5 du chapitre 2. Nous rappelons que cet étude de cas représente un ensemble de N clients qui essayent de télécharger un fichier partitionné en K blocs. Le protocole se termine lorsque tous les clients ont réussi à télécharger tous les blocs. Une description en B événementiel mixte de ce protocole est présentée dans la machine $\mathsf{P2P}_m$ en figure 4.8. Cette machine posséde les mêmes variables, le même invariant ainsi que les mêmes événements que les machines $\mathsf{P2P}_3$ et $\mathsf{P2P}_p$. La seule différence concerne le détail des événements DLFinished et Start1DL : ces deux événements ne possédent pas de poids, les choix des valeurs des paramètres est non-déterministe et les actions sont déterministes. Il s'agit donc d'événements standard alors que les événements Finish1DL et FailureDL sont deux événements probabilistes : les deux sont annotés par des poids, le choix des paramètres est uniforme et l'action de FailureDL est probabiliste.

Dans la description informelle de ce protocole, il est mentionné que plus le nombre de téléchargements effectués augmente, plus la chance qu'une erreur de téléchargement arrive diminue. Nous avons décidé d'annoter chacun des événements Finish1DL et FailureDL par un poids pour satisfaire cette contrainte. En effet, en regardant de près les gardes de ces deux événements, nous constatons visiblement que ces deux événements sont toujours activables simultanément. Le poids de l'événement Finish1DL est :

$$\mathsf{card}(\{\mathsf{c} \mapsto \mathsf{b} \mid \mathsf{c} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\mathsf{c} \mapsto \mathsf{b}) = \mathsf{finished}\}) + 1$$

Il représente le nombre de téléchargements effectués.
Le poids de l'événement FailureDL est :

$$\mathsf{N} \times \mathsf{K} - \mathsf{card}(\{\mathsf{c} \mapsto \mathsf{b} \mid \mathsf{c} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\mathsf{c} \mapsto \mathsf{b}) = \mathsf{finished}\})$$

Ce poids représente le nombre de téléchargements qui ne sont pas encore effectués.

Nous déduisons ainsi que plus le nombre de téléchargements effectués augmente, plus le poids de Finish1DL augmente, et plus le poids de l'événement FailureDL diminue, c'est-à-dire, la probabilité d'exécuter l'événement Finish1DL augmente et la probabilité d'exécuter l'événement FailureDL diminue.

MACHINE
P2P$_M$
VARIABLES
DB
DBin
INVARIANTS
DB$\in$ 1..N $\times$ 1..K $\rightarrow$ {empty,finished} $\wedge$ DBin $\in$ 1..N $\times$ 1..K $\rightarrow$ {empty,incoming,finished} $\wedge$
$\forall$ c . (c $\in$ 1..N $\Rightarrow$ card({b $\mid$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming}) $\leq$ 1)
VARIANT
2 $\times$ N $\times$ K $-$ 2 $\times$ card({c$\mapsto$b |c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = finished})
$-$ card({c$\mapsto$b $\mid$ c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming})
EVENTS
Init $\widehat{=}$
begin
DB := (1..N $\times$ 1..K) $\times$ {empty} $\|$
DBin := (1..N $\times$ 1..K) $\times$ {empty}
end
DLFinished $\widehat{=}$
when
DB = (1..N $\times$ 1..K) $\times$ {empty} $\wedge$ DBin = (1..N $\times$ 1..K) $\times$ {finished}
then
DB := DBin
end
Start1DL $\widehat{=}$
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = empty $\wedge$ card({k $\mid$ k $\in$ 1..K $\wedge$ DBin(c$\mapsto$k) = incoming})=0
then
DBin(c$\mapsto$b) := incoming
end
Finish1DL $\widehat{=}$
weight
card({c$\mapsto$b $\mid$ c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = finished}) +1
any c, b **where** c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming
then
DBin(c$\mapsto$b) := finished
end
FailureDL $\widehat{=}$
weight
N $\times$ K $-$ card({c$\mapsto$b |c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = finished})
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming
then
DBin(c$\mapsto$b) := empty @4/10 $\oplus$ incoming @6/10
end
END

FIGURE 4.8 – Une description du protocole P2P en B événementiel mixte

Sémantique Nous présentons dans la figure 4.5 les étapes initiales de la construction de l'automate probabiliste correspondant à la machine P2P$_m$ présentée dans la figure 4.8. Pour simplifier la construction, nous fixons le nombre de clients N ainsi que le nombre de blocs K à 2.

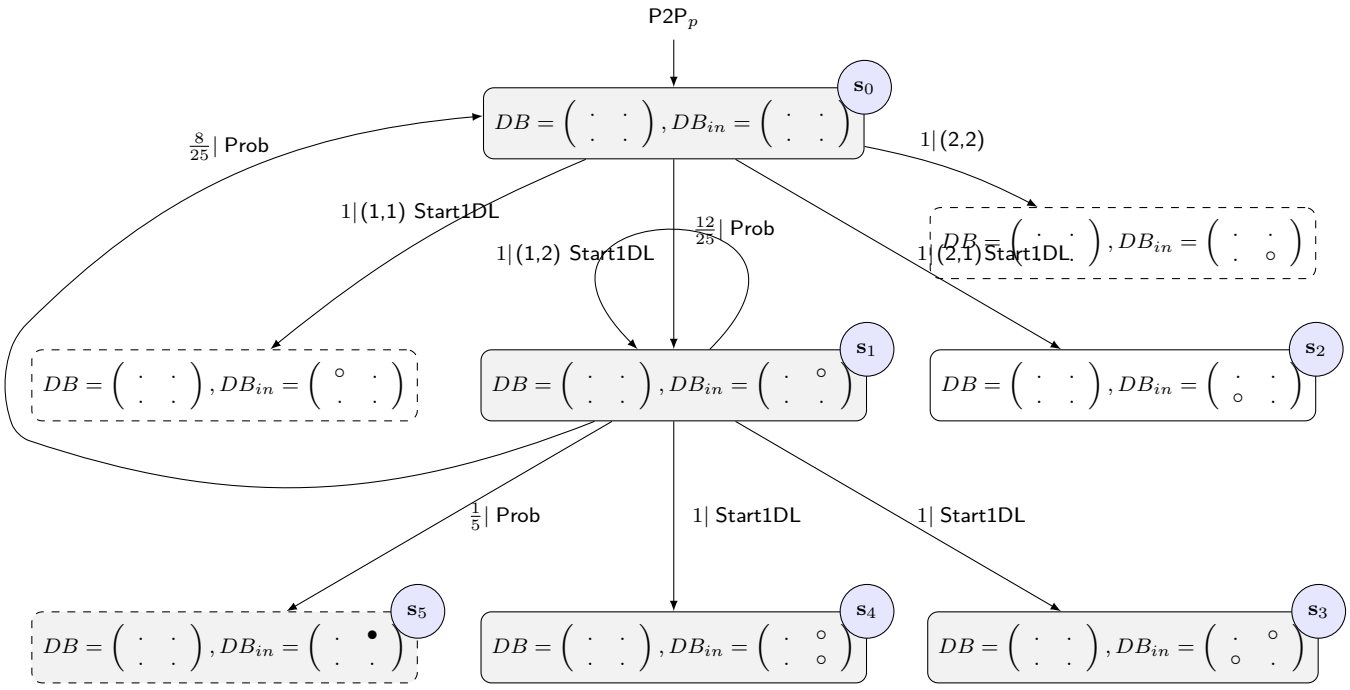

FIGURE 4.9 – Extrait de l'automate probabiliste du protocole P2P avec N=2 and K=2 (figure à modifier)

Dans l'état initial s_0, l'événement Start1DL est l'unique événement activable. Cet événement mène vers quatres distributions différentes de probabilité. Étant donné que l'événement Start1DL est non-déterministe, le choix entre ces distributions est un choix non-déterministe.

Dans l'état s_1, trois événements sont activables, FailureDL, Start1DL et Finish1DL. Nous rappelons que les événements FailureDL et Finish1DL sont probabilistes alors que l'événement Start1DL est non-déterministe.

À partir de cet état, nous remarquons que cinq transitions sont exécutables, deux transitions sont annotées par $Start1DL$ et trois sont annotées par $Prob$. Le choix entre les deux transitions non-déterministes et les trois transitions annotées par $Prob$ est non-déterministe. Chaque transition étiquetée par Start1DL est annotée par une valeur de probabilité de 1, c'est-à-dire qu'elle définit une distribution correcte de probabilité. Trois transitions probabilistes chacune étiquetée par $Prob$ sont aussi activables dans cet état. En effet, deux événements probabilistes sont activables dans s_1, Finish1DL et FailureDL. Nous regroupons les transitions correspondantes à ces événements sous l'étiquette $Prob$. Nous expliquons le calcul de ces probabilités de ces transitions dans ce qui suit.

- Nous considérons la transition qui mène vers l'état s_5, cette transition correspond à l'exécution de l'événement Finish1DL. Le poids de cet événement est :

$$\mathsf{card}(\{\mathsf{c}{\mapsto}\mathsf{b} \mid \mathsf{c} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\mathsf{c}{\mapsto}\mathsf{b}) = \mathsf{finished}\}) + 1$$

  Dans l'état s_1, aucun client n'a encore commencé le téléchargement d'un bloc, donc ce poids prend la valeur 1. Comme dans le cas probabiliste, la probabilité d'exécuter cet événement dans cet état correspond au ratio de son poids divisée par la somme des poids des autres événements activés dans cet état intermédiaire, cette probabilité est donc $\frac{1}{4+1} = \frac{1}{5}$.

- Nous considérons la transition qui boucle sur l'état s_1, cette transition correspond à une exécution de l'événement FailureDL où incoming est affecté à DBin avec une probabilité de $\frac{6}{10}$. Le poids de cet événement est

$$\mathsf{N} \times \mathsf{K} - \mathsf{card}(\{\mathsf{c} \mapsto \mathsf{b} \mid \mathsf{c} \in 1..\mathsf{N} \wedge \mathsf{b} \in 1..\mathsf{K} \wedge \mathsf{DBin}(\mathsf{c} \mapsto \mathsf{b}) = \mathsf{finished}\})$$

  Dans s_1, ce poids prend la valeur 4. La probabilité d'exécuter cet événement correspond au ratio de son poids divisé par la somme des poids des autres événements $\frac{4}{4+1} = \frac{4}{5}$. La probabilité ainsi de boucler sur l'état s_1 est $\frac{4}{5} \times \frac{6}{10} = \frac{12}{25}$

- Nous considérons la transition qui mène vers l'état s_0, cette transition correspond à une exécution de l'événement FailureDL où empty est choisie. Comme expliqué précédemment, la probabilité d'excuter cet événement est $\frac{4}{5}$. La probabilité de cette transition est $\frac{4}{5} \times \frac{4}{10} = \frac{16}{50}$

Nous remarquons finalement que la somme des transitions étiquetée par $Prob$ fait bien 1, ce qui définit une distribution correcte de probabilité.

5

Raffinements probabilistes

5.1 Introduction

Nous rappelons que notre objectif global consiste à introduire des probabilités en B événementiel. Nous avons présenté précédemment notre proposition de B événementiel purement probabiliste et de B événementiel mixte. Dans ce chapitre, nous proposons des solutions pour construire de telles machines dans un processus de développement B événementiel. Classiquement, ce processus est basé sur le raffinement. Dans [78], les auteurs ont mentionné que la solution la plus efficace pour intégrer des probabilités dans B événementiel consiste à raffiner (remplacer) toutes les sources de non-déterminisme par des probabilités. Dans ce chapitre, nous présentons plusieurs approches de développement permettant d'intégrer du raisonnement probabiliste en B événementiel. Nous reprenons dans la figure 5.1 la figure présentée dans l'introduction de ce manuscrit. Nous rappelons que cette figure présente plusieurs possibilités de développement qui permettent d'introduire des probabilités au sein du B événementiel. Dans cette figure, l'axe vetical représente l'introduction de plus de détails dans le modèle alors que l'axe horizontal représente l'introduction des probabilités. Plusieurs types de machines apparaissent dans cette figure, nous les présentons dans ce qui suit :

- Les machines de l'axe vertical de gauche sont des machines B événementiel standards qui contiennent uniquement des choix non-déterministes et donc aucune information probabiliste (notée Machine non-dét) ;
- Les machines de l'axe vertical de droite sont des machines B événementiel purement probabilistes où tous les choix sont probabilistes (et donc, plus aucun non-déterminisme) (notée Machine proba) ;
- Les machines dans le centre de la figure sont des machines B événementiel mixtes qui contiennent à la fois du non-déterminisme et des probabilités (notée Machine mixte).

En fonction du type du système à modéliser, le processus de développement peut rester toujours dans l'axe vertical de gauche (lorsque le système n'a aucun aspect probabiliste), et se terminer à la fin de cet axe. Le processus de développement peut aussi passer vers la partie droite et se termine à la fin de l'axe vertical de droite lorsque le système est purement probabiliste ou au milieu lorsque le système contient à la fois des aspects probabilistes et non-déterministes.

Plusieurs possibilités (chemins) de développement permettent l'ajout d'informations non-déterministes ou probabilistes.

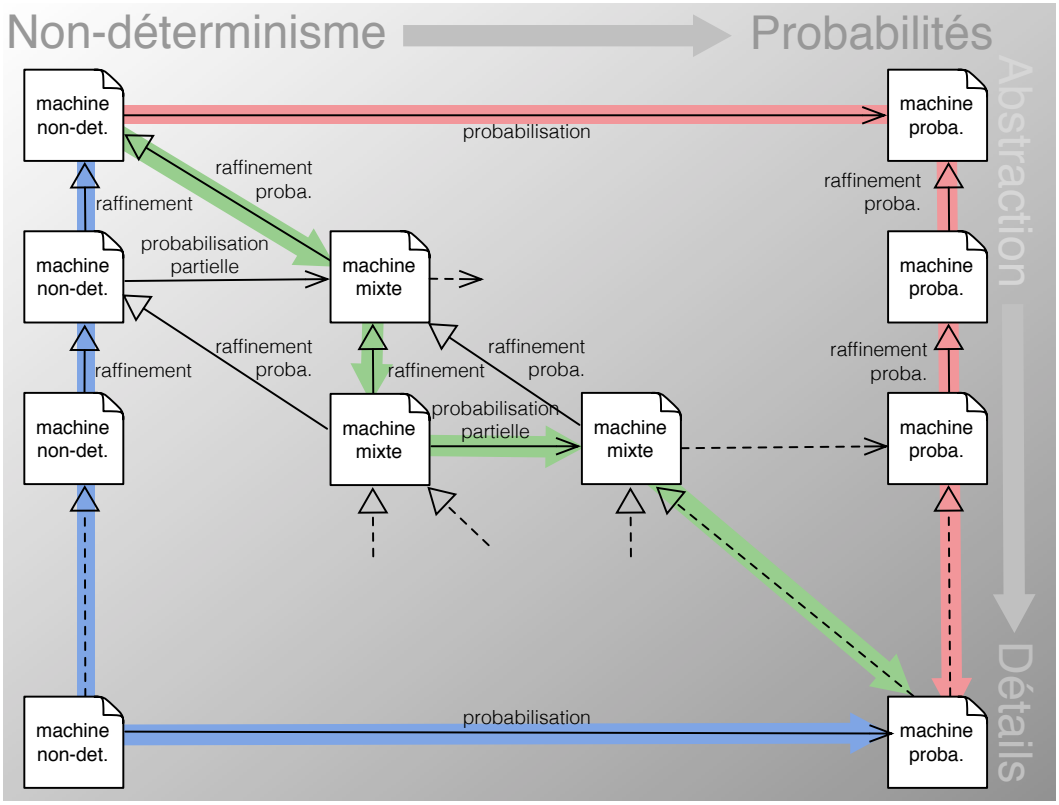

FIGURE 5.1 – Comment introduire des probabilités au sein du B événementiel ?

Supposons par exemple que le système à développer soit purement probabiliste, nous pouvons commencer à partir d'une machine abstraite non-déterministe, et progressivement, la raffiner par l'ajout de nouveaux détails jusqu'à l'obtention d'une description complète du système par cette machine. Finalement, nous remplaçons tous les choix non-déterministes de cette machine par des choix probabilistes, cette dernière étape s'appelle *la probabilisation*, elle est présentée par l'axe horizontal bleu en bas.

Une autre possibilité consiste à partir d'une machine non-déterministe abstraite, et appliquer la *probabilisation* sur cette machine obtenant ainsi une machine abstraite purement probabiliste et après, ajouter des détails probabilistes par raffinement probabiliste comme montré dans l'axe vertical rouge de droite. Une dernière possibilité consiste à ajouter simultanément des informations non-déterministes et des informations probabilistes obtenant ainsi des machines B événementiel mixtes (présentée par l'axe vert), par *probabilisation partielle* entre autres.

L'application de l'une ou l'autre de ces approches de développement dépend de type système à modéliser et du niveau du système le comportement probabiliste apparaît.

Pour récapituler, l'introduction des probabilités dans une machine B événementiel peut avoir lieu de plusieurs façons :

- La *probabilisation* totale consiste à remplacer tous les choix non-déterministes dans une machine B événementiel non-déterministe par des informations probabilistes, obtenant ainsi une machine B événementiel purement probabiliste ;
- La *probabilisation partielle* consiste à transformer un sous-ensemble d'événements non-déterministes en un sous-ensemble d'événements probabilistes par remplacement de tous les choix non-déterministes par des probabilités à l'intérieur de cet ensemble d'événements. La machine obtenue est une machine B événementiel mixte ;
- L'ajout par raffinement d'événements probabilistes ou déterministes dans une machine B

événementiel non-déterministe ou probabiliste. La machine obtenue est soit une machine B événementiel mixte ou probabiliste.

Dans ce chapitre, nous présentons chacune de ces approches de développement. Dans la section 5.2, nous présentons le processus de probabilisation (partielle ou totale). Dans la section 5.3, nous traitons l'introduction de nouveaux événements probabilistes dans une machine B événementiel probabiliste. Comme en B événementiel, introduire de nouveaux événements par raffinement nécessite la preuve de convergence de ces événements, nous traitons dans cette section la convergence d'un ensemble d'événements probabilistes. Dans la section 5.4, nous traitons le cas général d'introduction d'événements probabilistes ou non-déterministes dans une machine B événementiel de n'importe quel type.

5.2 Probabilisation

Dans cette section, nous présentons une approche consistant à transformer un ensemble d'événements standard en un ensemble d'événements probabilistes. Nous appelons ce processus *probabilisation*.

Le résultat de cette opération peut être une machine B événementiel probabiliste ou mixte. Ce résultat dépend principalement de l'ensemble d'événements probabilisés : si nous appliquons cette approche sur tous les événements d'une machine, alors la machine obtenue est une machine purement probabiliste. Si nous appliquons la probabilisation sur un sous ensemble d'événements d'une machine, alors la machine obtenue est une machine mixte.

La machine obtenue suite à l'application de ce processus possède les mêmes éléments (contextes, variables, invariants, variants..) que la machine initiale à laquelle ce processus est appliqué. Nous ne permettons pas l'ajout de nouvelles variables, de nouveaux invariants, de nouveaux contextes, de nouveaux variants ou de nouveaux événements. Chaque événement de la machine initiale garde sa structure, nous ne permettons ni l'ajout de nouveaux paramètres, ni la modification de la garde, ni l'ajout de nouvelles substitutions. La seule différence entre les deux machines consiste en l'ajout de poids pour tous les événements et la transformation des substitutions non-déterministes en substitutions probabilistes. Le processus de probabilisation est une sorte de raffinement simplifié, une machine M' probabilise une machine M si est seulement si :

1. La machine M' possède les mêmes variables $\bar{v}$ que la machine M,
2. La machine M' possède le mêmes invariant $I(\bar{v})$ que la machine M,
3. La machine M' voit les mêmes contextes $\overline{ctx}$ que la machine M,
4. La machine M' possède le même événement d'initialisation que la machine M,
5. À l'exception de l'événement d'initialisation, chaque événement de la machine M est contenu dans l'ensemble d'événements de la machine M' et modifié comme suit :
   (a) Chaque événement de la machine M est annoté par un poids dans la machine M',
   (b) Chaque substitution énumérée non-déterministe qui apparaît dans l'action d'un événement de la machine M est remplacée par une substitution énumérée probabiliste dans le même événement dans la machine M',
   (c) Chaque substitution non-déterministe sous forme de prédicat qui apparaît dans l'action d'un événement de la machine M est remplacée par une substitution probabiliste sous forme de prédicat dans le même événement dans la machine M'.

Notons que nous n'imposons aucune condition sur les valeurs des probabilités des substitutions.

5.2.1 Obligations de preuve de faisabilité de la probabilisation

Afin de pouvoir appliquer la probabilisation sur une machine B événementiel classique, cette dernière doit satisfaire certaines conditions permettant d'assurer que la machine probabiliste obtenue soit correcte. Ces conditions sont formalisées par des obligations de preuve, nous les présentons dans ce qui suit.

Probabilisation des paramètres Afin de pouvoir définir une distribution uniforme discrète sur l'ensemble des valeurs des paramètres de chaque événement e_i de M, cet ensemble doit être fini :

evt/param/proba

$\mathsf{I}(\bar{v}) \wedge$

$\vdash$

finite $(\{\bar{t} \mid \mathsf{G}_i(\bar{t},\bar{v})\})$

Probabilisation des événements Pour que nous puissions appliquer la probabilisation à un événement standard e_i, les substitutions de son action doivent satisfaire certaines conditions. En fonction du type de la substitution, une obligation de preuve spécifique est appliquée, nous les détaillons dans ce qui suit.

1. **Probabilisation d'une substitution non-déterministe énumérée.** Pour chaque substitution non-déterministe énumérée $\mathsf{x} :\in \{\mathsf{E}_1(\bar{t},\bar{v}),\ldots, \mathsf{E}_i(\bar{t},\bar{v}),\ldots, \mathsf{E}_n(\bar{t},\bar{v})\}$ d'un événement non-déterministe e_i de M, afin de pouvoir définir une distribution discrète de probabilités sur l'ensemble des expressions $\{\mathsf{E}_1(\bar{t},\bar{v}),\ldots, \mathsf{E}_i(\bar{t},\bar{v}),\ldots, \mathsf{E}_n(\bar{t},\bar{v})\}$, cet ensemble doit être fini :

evt/EnumSub/proba

$\vdash$

finite $(\mathsf{E}_1(\bar{t},\bar{v}),\ldots,\mathsf{E}_i(\bar{t},\bar{v}),\ldots,\mathsf{E}_n(\bar{t},\bar{v}))$

2. **Probabilisation d'une substitution non-déterministe sous forme de prédicat.** Afin de pouvoir définir une distribution uniforme discrète sur l'ensemble des valeurs x' qui rendent le prédicat $\mathsf{Q}_\mathsf{x}(\bar{t},\bar{v},\mathsf{x},\mathsf{x}')$ satisfait, cet ensemble doît être fini :

evt/PredSub/proba

$\mathsf{I}(\bar{v}) \wedge$

$\mathsf{G}_i(\bar{t},\bar{v})$

$\vdash$

finite $(\mathsf{x}' \mid \mathsf{Q}_\mathsf{x}(\bar{t},\bar{v},\mathsf{x},\mathsf{x}'))$

Lorsque toutes les conditions présentées ci-dessus sont satisfaites, l'ensemble des événements de la machine M peut être probabilisé.

5.2.2 Automatisation du processus de probabilisation

Nous notons également qu'il est possible de générer (*semi-automatiquement*) une machine B événementiel probabiliste par probabilisation d'une machine B événementiel non-déterministe, ce processus consiste à :

1. Générer une nouvelle machine B événementiel qui probabilise la machine M. Cette machine est purement probabiliste. Comme les variables, les invariants et les contextes sont non modifiées par la probabilisation, ils seront automatiquement inclus dans la machine B événementiel générée.
2. Probabiliser chaque événement de la machine initiale, c'est-à-dire :
   (a) Copier chaque événement de la machine initiale dans la machine B événementiel générée,
   (b) Annoter chaque événement e_i par une expression de poids $\mathsf{W}_i(\bar{v})$.
   (c) Remplacer chaque substitution non-déterministe énumérée par une substitution probabiliste énumérée de la forme :

$$\boxed{\mathsf{x} := \mathsf{E}_1(\bar{t},\bar{v}) \ @\ \mathsf{p}_1 \oplus \ldots \oplus \mathsf{E}_i(\bar{t},\bar{v}) \ @\ \mathsf{p}_i \oplus \ldots \oplus \mathsf{E}_n(\bar{t},\bar{v}) \ @\ \mathsf{p}_n}$$

   (d) Remplacer chaque substitution non-déterministe sous forme de prédicat par une substitution probabiliste sous forme de prédicat :

$$\boxed{\mathsf{x} :\oplus \mathsf{Q}_\mathsf{x}(\bar{t},\bar{v},\mathsf{x},\mathsf{x}')}$$

Par défaut, le poids prend la valeur 1, mais il peut aussi ensuite être précisé par le développeur. De même, les valeurs de probabilité p_i prennent par défaut la valeur $\frac{1}{n}$, où n est le nombre d'expressions dans la substitution. Ces valeurs peuvent être précisées par le développeur.

Note 4 *Il est possible d'appliquer la probabilisation sur un sous ensemble d'événements d'une machine standard et non sur tous ses événements. Nous appelons ce processus la probabilisation partielle. Toutefois, les événements de cet ensemble doivent satisfaire les obligations de preuve présentées ci-dessus.*

5.2.3 Étude de cas

Application du processus de probabilisation partielle Afin d'illustrer et d'expliquer le processus de probabilisation partielle, nous reprenons ici notre cas d'étude du protocole pair-à-pair. La machine $\mathsf{P2P}_3$ présentée dans la figure 2.10 représente une description détaillée du protocole. Sur cette machine, nous appliquons la probabilisation partielle sur le sous ensemble d'événements formé seulement des événements Finish1DL et FailureDL. Pour que l'on puisse probabiliser ces deux événements, nous devons nous assurer qu'ils satisfont les obligations de preuve de faisabilité de probabilisation. Finish1DL et FailureDL possèdent des paramètres. Pour que l'on puisse probabiliser chacun de ses événements, nous devons nous assurer dans un premier temps que l'ensemble des valeurs des paramètres de chacun de ces événements est fini ((evt/param/proba)). Chacun de ces événements possède deux paramètres c et b tel que $\mathsf{c} \in 1..\mathsf{N}$ et $\mathsf{b} \in 1..\mathsf{K}$. Puisque N et K sont bornés, nous déduisons ainsi que l'ensemble des valeurs des paramètres de chacun de ces événements est fini. L'événement FailureDL posséde une substitution non-déterministe énumérée $\mathsf{DBin}(\mathsf{c}\mapsto\mathsf{b}) :\in \{\mathsf{empty},\mathsf{incoming}\}$. Évidemment, l'ensemble des expressions de cette substitution est fini et ainsi l'obligation de preuve (evt/EnumSub/proba) est remplie pour cet événement.

Ces deux événements satisfont les obligations de preuve de faisabilité de probabilisation. Ainsi, nous pouvons les probabiliser. Nous obtenons alors la machine $\mathsf{P2P}_m$ présentée dans la figure 5.2.

MACHINE
P2P$_m$
PROBABILISES
P2P$_3$
VARIABLES
DB
DBin
INVARIANTS
DB $\in$ 1..N $\times$ 1..K $\rightarrow$ {empty,finished} $\wedge$
DBin $\in$ 1..N $\times$ 1..K $\rightarrow$ {empty,incoming,finished} $\wedge$
$\forall$ c . (c $\in$ 1..N $\Rightarrow$ card({b $\mid$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming}) $\leq$ 1)
VARIANT
2 $\times$ N $\times$ K $-$ 2 $\times$ card({c$\mapsto$b $\mid$ c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = finished})
$-$ card({c$\mapsto$b $\mid$ c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming})
EVENTS
Init $\widehat{=}$
begin
DB := (1..N $\times$ 1..K) $\times$ {empty} ||
DBin := (1..N $\times$ 1..K) $\times$ {empty}
end
DLFinished $\widehat{=}$
when
DB = (1..N $\times$ 1..K) $\times$ {empty} $\wedge$
DBin = (1..N $\times$ 1..K) $\times$ {finished}
then
DB := DBin
end
Start1DL $\widehat{=}$
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = empty $\wedge$
card({k $\mid$ k $\in$ 1..K $\wedge$ DBin(c$\mapsto$k) = incoming})=0
then
DBin(c$\mapsto$b) := incoming
end
Finish1DL $\widehat{=}$
weight
card({c$\mapsto$b $\mid$ c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = finished}) +1
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming
then
DBin(c$\mapsto$b) := finished
end
FailureDL $\widehat{=}$
weight
N $\times$ K $-$ card({c$\mapsto$b $\mid$ c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = finished})
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming
then
DBin(c$\mapsto$b) := empty @ 4/10 $\oplus$ incoming @ 6/10
end
END

FIGURE 5.2 – Une machine B événementiel mixte du protocole P2P

Dans la description informelle de ce protocole, il est mentionné que plus le nombre de téléchargements effectués augmente, plus la chance qu'une erreur de téléchargement arrive diminue. Nous annotons chacun des événements Finish1DL et FailureDL par un poids exprimant cette contrainte.

Nous annotons l'événement Finish1DL par un poids qui représente le nombre de téléchargements effectués, ce poids est donné par :

$$\mathsf{card}(\{c{\mapsto}b \mid c \in 1..N \land b \in 1..K \land \mathsf{DBin}(c{\mapsto}b) = \mathsf{finished}\})+1$$

Nous annotons l'événement FailureDL par un poids qui représente le nombre de téléchargements qui ne sont pas encore effectués. Ce poids est donné par :

$$N \times K - \mathsf{card}(\{c{\mapsto}b \mid c \in 1..N \land b \in 1..K \land \mathsf{DBin}(c{\mapsto}b) = \mathsf{finished}\})$$

Ainsi, plus le nombre de téléchargements effectués augmente, plus le poids de Finish1DL augmente et plus le poids de l'événement FailureDL diminue, c'est-à-dire, la probabilité d'exécuter l'événement Finish1DL augmente et la probabilité d'exécuter l'événement FailureDL diminue.

Application du processus de probabilisation Nous reprenons ici le même cas d'étude du protocole pair-à-pair et nous appliquons la probabilisation sur l'ensemble des événements de la machine. Les événements Finish1DL et FailureDL sont probabilisés comme précédemment. Concernant DLFinished et Start1DL, ils peuvent également être probabilisés : l'ensemble des paramètres de l'événement Start1DL est fini, DLFinished n'a pas de paramètres et son action est déterministe. Nous obtenons ainsi la machine $\mathsf{P2P}_p$ présentée dans la figure 5.3.

MACHINE
$\mathsf{P2P}_p$
VARIABLES
DB
DBin
INVARIANTS
$\mathsf{DB} \in 1..N \times 1..K \rightarrow \{\mathsf{empty},\mathsf{finished}\} \land$
$\mathsf{DBin} \in 1..N \times 1..K \rightarrow \{\mathsf{empty},\mathsf{incoming},\mathsf{finished}\} \land$
$\forall c \cdot (c \in 1..N \Rightarrow \mathsf{card}(\{b \mid b \in 1..K \land \mathsf{DBin}(c{\mapsto}b) = \mathsf{incoming}\}) \leq 1)$
VARIANT
$2 \times N \times K - 2 \times \mathsf{card}(\{c{\mapsto}b \mid c \in 1..N \land b \in 1..K \land \mathsf{DBin}(c{\mapsto}b) = \mathsf{finished}\})$
$- \mathsf{card}(\{c{\mapsto}b \mid c \in 1..N \land b \in 1..K \land \mathsf{DBin}(c{\mapsto}b) = \mathsf{incoming}\})$
EVENTS
Init $\widehat{=}$
begin
$\mathsf{DB} := (1..N \times 1..K) \times \{\mathsf{empty}\} \parallel$
$\mathsf{DBin} := (1..N \times 1..K) \times \{\mathsf{empty}\}$
end
DLFinished $\widehat{=}$
weight $N \times K$
when
$\mathsf{DB} = (1..N \times 1..K) \times \{\mathsf{empty}\} \land$
$\mathsf{DBin} = (1..N \times 1..K) \times \{\mathsf{finished}\}$
then
$\mathsf{DB} := \mathsf{DBin}$
end

.

```
Start1DL ≙
weight
 N × K − card({c↦b | c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = incoming})
any c, b where
 c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = empty ∧
 card({k | k ∈ 1..K ∧ DBin(c↦k) = incoming})=0
then
 DBin(c↦b) := incoming
end
Finish1DL ≙
weight
card({c↦b | c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = finished}) +1
any c, b where
 c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = incoming
then
 DBin(c↦b) := finished
end
FailureDL ≙
weight
 N × K − card({c↦b |c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = finished})
any c, b where
 c ∈ 1..N ∧ b ∈ 1..K ∧ DBin(c↦b) = incoming
then
 DBin(c↦b) := empty @4/10 ⊕ incoming @6/10
end
```

FIGURE 5.3 – Une machine B événementiel purement probabiliste du protocole P2P

Après avoir introduit les approches de probabilisation et de probabilisation partielle, nous introduisons dans la section suivante l'introduction de probabilités en B événementiel par ajout de nouveaux événements probabilistes.

5.3 Introduction de nouveaux événements par raffinement

Le processus de développement en B événementiel est basé sur le raffinement. L'un des principaux aspects du raffinement en B événementiel consiste en l'ajout de nouveaux événements comme présenté dans la section 2.4.3. Dans cette section, nous traitons le raffinement d'une machine B événementiel probabiliste et nous nous concentrons sur une variante spécifique du raffinement : l'ajout d'événements probabilistes.

Note 5 *Dans cette section, nous notons par convergent un événement qui est introduit par raffinement dans une machine B événementiel et qui doit converger.*

En B événementiel, lorsque l'on ajoute de nouveaux événements, nous devons montrer que leur introduction n'empêche pas le système de se comporter comme spécifié dans la machine abstraite. Cela revient à montrer que l'ensemble de ces nouveaux événements introduits par raffinement converge, c'est-à-dire que les événements de cet ensemble ne s'exécute pas indéfiniment. À un certain moment, le système doit arrêter d'exécuter de nouveaux événements, afin de revenir au comportement spécifié dans la machine abstraite. Afin d'assurer cette propriété de convergence en B événementiel, un variant de type entier est proposé et toutes les valuations des variables obtenues après l'exécution de l'action de chaque nouvel événement doivent décroître la valeur de ce variant. Dans le cadre probabiliste, nous ne nous intéressons pas à la convergence mais

plutôt à la convergence presque certaine. En effet, lorsqu'il sagit de nouveaux événements probabilistes introduits, nous ne prouverons pas que l'ensemble de nouveaux événements converge mais nous prouverons que cet ensemble converge *presque certainement* [57, 75], c'est-à-dire converge avec une probabilité de 1. Prouver la convergence *presque certaine* d'un ensemble d'événements probabilistes revient à prouver que la probabilité d'exécuter infiniment ces nouveaux événements probabilistes est de 0. Autrement dit, ceci consiste à prouver que dans tous les états du système où de nouveaux événements probabilistes convergent sont activables, la probabilité d'exécuter un événement non-convergent ou atteindre un état de blocage est de 1.

Dans cette section, nous revenons sur les résultats des travaux de recherche ayant déjà traité la convergence presque certaine d'un ensemble d'événements en B événementiel. Par la suite, nous présentons notre proposition comme une adaptation des travaux précédents pour traiter la convergence *presque certaine* d'un ensemble d'événements probabilistes.

5.3.1 Convergence presque certaine en B événementiel dans la littérature

Comme mentionnés dans la section 2.7, les travaux de Hallerstede et Hoang [54, 58] ont introduit un nouveau type de raisonnement en B événementiel appelé *Raisonnement probabiliste qualitatif*. Dans cette proposition, les auteurs ont introduit une nouvelle substitution appelée substitution probabiliste qualitative qui s'écrit :

$$x :\oplus \mathsf{Q}_x(\bar{t},\bar{v},x,x')$$

Dans cette substitution, la variable x prend une nouvelle valeur x' avec une probabilité strictement positive tel que le prédicat $\mathsf{Q}_x(\bar{t},\bar{v},x,x')$ sera satisfait. L'objectif de ces travaux est d'introduire le raisonnement sur la convergence *presque certaine* d'un ensemble d'événements. Les auteurs ont considéré un ensemble d'événements dont les actions ne contiennent que des substitutions déterministes ou probabilistes qualitatives. Les auteurs ont étudié la convergence presque certaine de cet ensemble et ont formalisé des conditions nécessaires pour prouver la convergence presque certaine par un ensemble d'obligations de preuve : ils ont adapté les obligations de preuve de convergence classiques du B événementiel et ont ajouté quelques nouvelles obligations de preuve.

- En B événementiel standard, l'obligation de preuve evt/VAR assure pour chaque événement qui doit converger que toutes les valuations des variables qui sont obtenues après l'exécution de son action doivent décroître la valeur du variant. Hoang et Hallerstede ont allégé cette contrainte, ils ont montré dans leur contexte qu'il suffit qu'au moins une valuation des variables obtenue après l'exécution de son action décroît la valeur du variant. Cela permet que certaines valuations des variables obtenues après l'exécution de l'action de l'événement peuvent accroître la valeur du variant ;
- Afin d'empêcher l'augmentation infinie de la valeur du variant, Hallerstede et Hoang ont imposé une nouvelle contrainte : ils ont introduit une borne maximale $\mathsf{U}(\bar{v})$ et ils ont imposé que la valeur du variant ne doive jamais dépasser cette borne maximale.

Toutes ces conditions ont été formalisées par des obligations de preuve que nous avons présenté dans la section 2.7.

5.3.2 Convergence presque certaine dans notre proposition de B événementiel probabiliste

Nous considérons une machine B événementiel complètement probabiliste, et nous introduisons un ensemble d'événements probabilistes dans cette machine. Dans cette section, nous étudions

l'ensemble des conditions permettant de prouver que cet ensemble des nouveaux événements probabilistes converge *presque certainement* (converge avec une probabilité 1). Nous formalisons ces conditions par un ensemble d'obligations de preuve. Nous adaptons les obligations de preuve proposées dans les travaux de Hallerstede et Hoang [54, 58] au cas des événements probabilistes introduits par notre proposition et nous introduisons de nouvelles obligations de preuve.

Nous rappelons que prouver la convergence presque certaine d'un ensemble d'événements probabilistes revient à prouver que, dans tous les états de la machine où au moins un de ces événements est activable, la probabilité d'exécuter un événement non-convergent ou d'atteindre un état de blocage est de 1 (la probabilité d'exécuter infiniment des événements probabilistes convergents est 0).

Dans notre proposition, nous avons remplacé le choix non-déterministe entre les événements activables dans une même valuation des variables par un choix probabiliste. Ce remplacement induit des modifications sur la condition de la décroissance du variant formalisée par l'obligation de preuve (evt/VAR) dans [54, 58]. En effet, lorsque le choix entre les événements activables dans la même valuation des variables est non-déterministe, nous devons montrer pour tous les événements activables dans cette valuation que chacun possède au moins une valuation des variables obtenue après son exécution qui fait décroître le variant. Lorsque le choix est probabiliste, chaque événement s'exécute avec une valeur de probabilité strictement positive. Il suffit alors que dans chaque valuation des variables où au moins un événement probabiliste est activable qu'il existe une valuation des variables obtenue après l'exécution de l'un de ces événements qui décroît la valeur du variant. Ainsi, il suffit de prouver qu'au moins un événement probabiliste possède une valuation des variables qui décroît la valeur du variant. De cette manière, la probabilité globale de décroitre le variant est strictement supérieure à zéro.

Nous simplifions ainsi l'obligation de preuve (evt/VAR) donnée en [54, 58] afin qu'elle prenne en compte cette nouvelle contrainte.

Afin d'empêcher l'augmentation infinie du variant, nous imposons comme dans [54] que le variant soit borné par une borne maximale. Afin de simplifier notre raisonnement, nous imposons que la borne supérieure soit une constante U et ne dépendant pas des variables de la machine. Nous expliquons plus loin comment étendre ce raisonnement à une borne non fixe $U(\bar{v})$ qui dépend des variables de la machine.

Nous présentons dans ce qui suit les oligations de preuve correspondantes à ces conditions.

Convergence presque-certaine Cette obligation de preuve assure que, dans toutes les valuations des variables où au moins un des événements probabilistes convergents est activable, il doit exister *au moins* une valuation des variables $\bar{v}'$ obtenue après l'exécution de l'un de ces événements qui décroît le variant (les événements de i+1 jusqu'à n sont convergents).

machine/pVar

$$I(\bar{v}) \wedge ((G_i(\bar{t},\bar{v}) \wedge W_i(\bar{v}) > 0) \vee \ldots \vee (G_n(\bar{t},\bar{v}) \wedge W_n(\bar{v}) > 0))$$

$$\vdash$$

$$(\exists\, \bar{v}'.\ G_i(\bar{t},\bar{v}) \wedge W_i(\bar{v}) > 0 \wedge SP_i(\bar{t},\bar{v},\bar{v}') \wedge V(\bar{v}') < V(\bar{v}))$$

$$\vee \ldots \vee .$$

$$(\exists\, \bar{v}'.\ G_n(\bar{t},\bar{v}) \wedge W_n(\bar{v}) > 0 \wedge SP_n(\bar{t},\bar{v},\bar{v}') \wedge V(\bar{v}') < V(\bar{v}))$$

Variant minoré Cette obligation de preuve assure que chaque événement probabiliste convergent e_i n'est activable que lorsque le variant est minoré.

evt/var/pNAT

$$I(\bar{v}) \wedge G_i(\bar{t}, \bar{v}) \wedge W_i(\bar{v}) > 0 \vdash V(\bar{v}) \in NAT$$

Variant majoré Cette obligation de preuve assure que chaque événement probabiliste convergent e_i n'est activable que lorsque le variant est majoré par une borne maximale U.

evt/pBOUND

$$I(\bar{v}) \wedge G_i(\bar{t}, \bar{v}) \wedge W_i(\bar{v}) > 0 \vdash V(\bar{v}) \leq U$$

Nous traitons des systèmes à nombre infini d'états, les obligations de preuve (evt/var/pNAT), (machine/pVar) et (evt/pBOUND) présentées ci-dessus ne sont pas suffisantes en général pour prouver que la probabilité d'exécuter fatalement un événement non-convergent ou d'atteindre un état de blocage est 1. Par définition d'une substitution probabiliste énumérée et d'une substitution probabiliste sous forme de prédicat, le nombre de valuations des variables qui peuvent être obtenues suite à l'exécution d'un événement probabiliste donné est toujours fini. Cependant, les poids des événements ainsi que les choix des paramètres peuvent rendre la probabilité de décroître le variant infinimement petite. Ce problème est dû au fait que les poids des événements probabilistes ainsi que les valeurs des paramètres ne sont pas bornés : si les valeurs de ces poids augmentent infiniment, la probabilité de décroitre le variant peut diminuer infiniment. Afin d'illustrer cet aspect, nous présentons dans ce qui suit deux exemples de machines B événementiel avec des poids non bornés et un nombre de valeurs de paramètres qui n'est pas fini dans lequelles nous allons montrer que la convergence presque certaine n'est pas assurée.

Exemple 1 : Nécessité de borner les poids des événements Nous montrons à travers un exemple d'une machine B événementiel probabiliste la nécessité de majorer les poids des événements probabilistes convergents afin d'assurer leur convergence presque certaine. Considérons la machine B événementiel probabiliste M1 présentée dans la figure 5.4.

Cette machine possède deux variables x et y et trois événements evt1, evt2 et evt3. Les événements evt1 et evt2 sont convergents. Le variant de cette machine est x et une borne possible de ce variant est clairement U=2. Un extrait de la sémantique opérationnelle de M_1 est présenté dans la figure 5.5. Dans chaque état de la figure, les valeurs sont données par (x, y).

MACHINE
M1
VARIABLES
x, y
INVARIANTS
x ∈ NAT ∧ y ∈ NAT
VARIANT
x
EVENTS
Init ≙
begin
x := 1
y := 2
end
evt1 ≙
convergent
weight 1
when 0 < x ≤ 2 **then**
x := x−1
y := 2∗y
end

evt2 ≙
convergent
weight y−1
when 0 <x ≤ 1 **then**
x := x+1
end

evt3 ≙
weight 1 **when** x = 0 **then**
x := −1
y := −1

end

END

FIGURE 5.4 – Machine B événementiel probabiliste M_1

En regardant la sémantique de M1, nous remarquons que dans les états où x=1, uniquement les événements convergents evt1 et evt2 sont activables, la probabilité de choisir l'événement evt1 dans ces états est $\frac{1}{y}$ alors que la probabilité de choisir l'événement evt2 est $\frac{y-1}{y}$. Dans les états où $x = 2$, uniquement l'événement evt1 peut être activable avec une probabilité 1. Dans les états où x=0, le seul événement activable est l'événement non convergent evt3.

Clairement, la machine M1 satisfait les obligations de preuve (machine/pVAR), (evt/var/pNAT) et (evt/var/pBOUND). Cependant, la probabilité de finir par exécuter fatalement l'événement non convergent evt3 est strictement inférieure à 1 à partir de tous les états où $x > 0$ car la probabilité de décroitre le variant diminue infiniment à partir des états où $x = 1$ lorsque y augmente. Cette probabilité reste positive dans tous les états. Nous calculons la probabilité de finir par choisir l'événement evt3 à partir de l'état initial où $x = 1$ et $y = 2$. Cette probabilité est égale à la somme de :

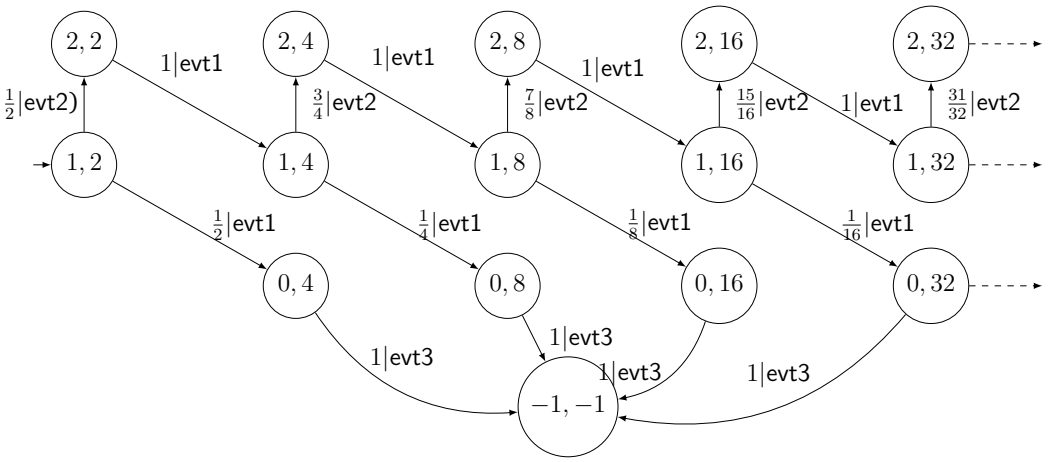

FIGURE 5.5 – Extrait de la sémantique opérationnelle de M1

(1) la probabilité d'exécuter l'événement evt1 à partir de l'état $(1, 2)$,

(2) la probabilité d'atteindre l'état $(1, 4)$ et exécuter l'événement evt1 à partir de l'état $(1, 4)$,

(3) la probabilité d'atteindre l'état $(1, 8)$ et exécuter l'événement evt1 à partir de l'état $(1, 8)$,

(4) ...

Clairement, (1) est égale à $\frac{1}{2}$, (2) est égale à $\frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$, (3) est égale à $\frac{1}{2} \cdot \frac{3}{4} \cdot \frac{1}{8} < \frac{1}{16}$ et généralement, la probabilité d'atteindre l'état $(1, 2^i)$ avec $i > 2$ et exécuter l'événement evt1 à partir de cet état est strictement inférieure à $\frac{1}{2^{i+1}}$.

En conséquence, la probabilité de fatalement exécuter l'événement evt3 à partir de l'état initial est strictement inférieure à

$$\frac{1}{2} + \sum_{i=2}^{\infty} \frac{1}{2^{i+1}} = \frac{3}{4}$$

Ainsi, M1 ne converge pas presque certainement.

Le problème montré ici est une conséquence directe du fait que les poids des événements convergents evt1 et evt2 ne sont pas bornés. Ainsi, si ces poids augmentent infiniment, la probabilité de décroître le variant peut diminuer infinimement et être de plus en plus petite.

Exemple 2 : nécessité de borner le nombre de valeurs de paramètres Nous présentons ici par l'intermédiaire d'une machine B événementiel probabiliste la nécessité de borner le nombre de valeurs des paramètres d'un événement probabiliste convergent. Nous présentons dans la figure 5.6 la machine M2, un extrait de sa sémantique correspondante exprimée en termes de chaîne de Markov discrète est présentée dans la figure 5.7.

MACHINE
M_2

VARIABLES
x
y
INVARIANTS
$x \in NAT$
$y \in NAT$
variant
x
EVENTS

Init $\widehat{=}$
then
$x := 1$
$y := 1$
end
evt1 $\widehat{=}$
convergent
weight 1
any t **where**
$t \in \{1..\ 2^y\} \wedge 0 < x \leq 1$
then
$x :\oplus ((t = 1 \wedge x' - x + t = 0)$ or
$(2 \leq t \leq 2^y \wedge x' - x - 1 = 0))$
$y := y + 1$
end

evt2 $\widehat{=}$
convergent
weight 1
when $x = 2$ **then**
$x := x - 1$
end

evt3 $\widehat{=}$
weight 1
when $x = 0$ **then**
$x := -1$
$y := -1$
end
END

FIGURE 5.6 – Machine B événementiel probabiliste M_2

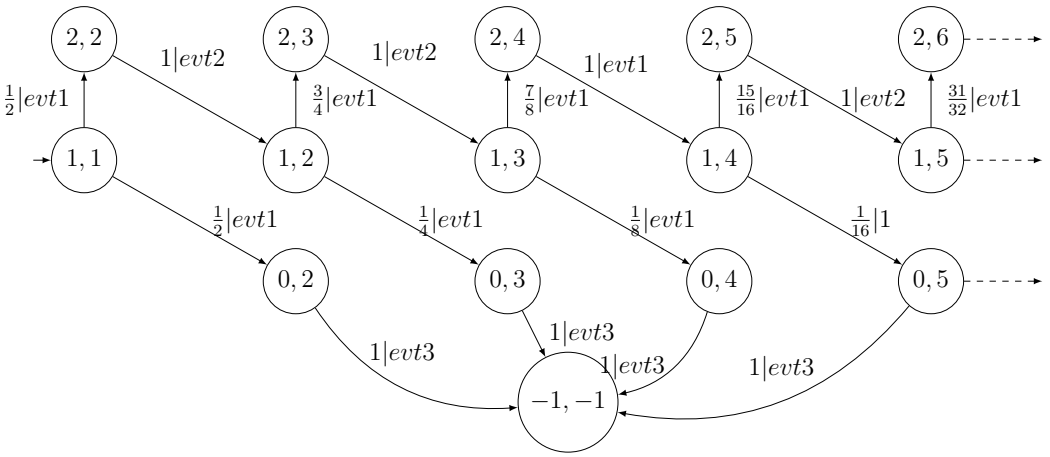

FIGURE 5.7 – Extrait de la sémantique opérationnelle de M_2

Cette machine ressemble à la machine de l'exemple précédent. Nous remarquons que la probabilité d'exécuter fatalement un événement non convergent evt3 à partir de l'état initial est strictement inférieur à $\frac{3}{4}$. Dans la machine M1, ce sont les poids des événements probabilistes convergents qui sont à l'origine de la décroissance infinie de la valeur de la probabilité de décroître le variant. C'est le choix des valeurs des paramètres des événements probabilistes convergents qui sont à l'origine de la décroissance infinie de la valeur de la probabilité de décroître le variant.

Les deux exemples donnés précédemment nous montrent la nécessité de borner les valeurs des poids et le nombre de valeurs des paramètres. Nous formalisons ainsi ces deux contraintes ajoutées sur les poids et les valeurs des paramètres par des nouvelles obligations de preuve.

Nouvelles obligations de preuve Nous formalisons les conditions sur les poids des événements probabilistes convergents et le nombre de valeurs des paramètres des événements probabilistes convergents présentées dans l'exemple 1 et l'exemple 2 par deux nouvelles obligations de preuve. Informellement, ces deux obligations de preuve assurent que la probabilité de décroitre le variant ne peut pas décroître infiniment en imposant que le poids de chaque événement probabiliste convergent soit borné par une constante BW et que le nombre des valeurs possibles des paramètres de chaque événement probabiliste convergent soit aussi borné par une constante BP. Afin de simplifier notre raisonnement, nous imposons que les bornes maximales soient des constantes. Nous expliquons plus loin dans ce manuscrit comment étendre notre raisonnement au cas des bornes variables BW($\bar{v}$) et BP($\bar{v}$). Nous présentons ces deux obligations de preuve dans ce qui suit.

- **Poids borné.** Cette obligation de preuve assure que la valeur du poids de chaque événement probabiliste convergent e_i est majoré par une constante BW.

evt/wght/BOUND
$\mathsf{I}(\bar{v}) \wedge$
$\mathsf{G}_i(\bar{t}, \bar{v})$
$\vdash$
$\mathsf{W}_i(\bar{v}) \leq \mathsf{BW}$

- **Nombre de valeurs des paramètres borné.** Cette obligation de preuve assure pour chaque événement probabiliste convergent e_i que le nombre de valeurs de ses paramètres est majoré par une constante BP.

evt/param/BOUND
$\mathsf{I}(\bar{v})$
$\vdash$
$\mathsf{card}(\{\bar{t} \mid \mathsf{G}_i(\bar{t},\bar{v})\}) \leq \mathsf{BP}$

Preuve de convergence. Nous montrons par la suite que les obligations de preuve présentées dans cette section sont suffisantes pour montrer la convergence *presque certaine* d'un ensemble d'événements probabilistes convergents donné.

Théorème 1 Soit M=($\bar{v}$,I($\bar{v}$),V($\bar{v}$),PEvts, Init) une machine B événementiel probabiliste et soit $\mathsf{PEvts}_c \subseteq \mathsf{PEvts}$ un ensemble d'événements probabilistes convergents. Si les événements de PEvts_c satisfont les obligations de preuve (evt/var/pNAT), (machine/pVar), (evt/var/pBOUND), (evt/wght/-BOUND) et (evt/param/BOUND), alors l'ensemble PEvts_c converge *presque certainement*.

Preuve

Soit M=($\bar{v}$,I($\bar{v}$),V($\bar{v}$),PEvts, Init) une machine B événementiel probabiliste. Soit $\mathsf{Evts}=\mathsf{Evts}_{nc} \cup \mathsf{Evts}_c$ la partition de l'ensemble d'événements Evts en le sous-ensemble d'événements non convergents $\mathsf{Evts}_{nc}= \mathsf{e}_1 \ldots \mathsf{e}_i$ et le sous-ensemble d'événements convergents $\mathsf{Evts}_c= \mathsf{e}_{i+1} \ldots \mathsf{e}_n$ ($1 \leq \mathsf{i} \leq \mathsf{n}$).

Nous allons montrer que si les événements de Evts_c satisfait les obligations de preuve rappelées ci-dessous, alors les événements de Evts_c convergent *presque certainement* (avec une probabilité 1).

1. evt/var/pNAT
$$\forall e \in \mathsf{Evts}_c.I(\bar{v}) \wedge W_e(\bar{v}) > 0 \wedge G_e(\bar{t},\bar{v}) \vdash V(\bar{v}) \in NAT$$
2. evt/pBOUND
$$\forall e \in \mathsf{Evts}_c.I(\bar{v}) \wedge W_e(\bar{v}) > 0 \wedge G_e(\bar{t},\bar{v}) \vdash V(\bar{v}) \leq U$$
3. evt/wght/BOUND
$$\forall e \in \mathsf{Evts}_c.I(\bar{v}) \wedge G_e(\bar{t},\bar{v}) \vdash W(\bar{v}) \leq BW$$
4. evt/param/BOUND
$$\forall e \in \mathsf{Evts}_c.I(\bar{v}) \vdash card(\{\bar{t}|G_e(\bar{t},\bar{v})\}) \leq BP$$
5. machine/pVar
$$I(\bar{v}) \wedge (G_{i+1}(\bar{t},\bar{v}) \vee ... \vee G_n(\bar{t},\bar{v})) \vdash (\exists \bar{v}'.W_{i+1}(\bar{v}) \wedge G_{i+1}(\bar{t},\bar{v}) \wedge SP_{i+1}(\bar{t},\bar{v}) \wedge V(\bar{v}') < V(\bar{v})) \vee ...$$
$$\vee(\exists \bar{v}'.W_n(\bar{v}) \wedge G_n(\bar{t},\bar{v}) \wedge SP_n(\bar{t},\bar{v}) \wedge V(\bar{v}') < V(\bar{v}))$$

Nous rappelons que prouver que Evts_c converge presque certainement consiste à prouver que dans toutes les valuations des variables de M où au moins un événement convergent est activable, la probabilité d'exécuter fatalement un événement non convergent ou d'atteindre un blocage est égale à 1.

Afin de prouver ce résultat, nous considérons une version modifiée de la sémantique de M et nous utilisons des résultats classiques sur les chaines de Markov discrètes infinies [74] afin de montrer que la probabilité d'atteindre un ensemble d'états donné à partir de tous les états où des événements non convergents sont activables est de 1.

Pour différencier l'ensemble des événements probabilistes convergents de l'ensemble des événements probabilistes non convergents, nous proposons une version modifiée de la sémantique de M. Dans cette nouvelle version modifiée, tous les états sont dupliqués afin de "mémoriser" le dernier événement exécuté.

Formellement, considérons la machine B événementiel probabiliste M=($\bar{v}$,I($\bar{v}$),V($\bar{v}$),PEvts, Init) et soit $[\![M]\!] = (S, s_0, \mathsf{Acts}, P)$ la sémantique correspondante à M. Nous construisons la chaîne de Markov discrète $[\![M]\!]' = (T, t_0, \mathsf{Acts}, P')$ où :

- $T \subseteq S \times (\mathsf{Acts} \cup \{\epsilon\})$ est l'ensemble des états étendus. Ces états sont des paires (s, a) où s est un état de $[\![M]\!]$ et a est une action (le nom d'un événement),
- $t_0 = (s_0, \epsilon)$ est l'état initial,
- P' est telle que $P'((s, a), e, (s', b)) = P(s, e, s')$ si $e = b$ et 0 dans le cas contraire pour toute action a. Nous rappelons que pour un événement probabiliste e_i, la fonction de probabilité P définie dans la section 3.5 est donnée par :

$$P(s, e_i, s') = \frac{[s]W_i(\bar{v})}{\sum_{e_j \in \mathsf{Acts}(s)} [s]W_j(\bar{v})} \times \sum_{\theta \in T_s^{e_i}} \left(P_{T_s^{e_i}}(\theta) \times \prod_{\mathsf{x} \in Var(e_i)} P_{s,\theta}^{e_i}(\mathsf{x}, s') \right)$$

Nous pouvons constater que Evts_c converge presque certainement si et seulement si la probabilité d'atteindre fatalement un état de blocage ou un état étendu de la forme $t = (s, e)$ où e est un événement non convergent est égale à 1 dans $[\![M]\!]'$ à partir de tous les états étendus où certains événements convergents sont activables.

Puisque $[\![M]\!]$ possède un ensemble infini d'états, montrer un tel résultat n'est pas simple. Afin de le prouver, nous exploitons des résultats classiques sur les chaines de Markov discrètes. Particulièrement, nous utilisons la propriété de *Global Coarseness* introduite par Mayr et *al.* [74].

Formellement, soit une chaîne de Markov discrète $\mathcal{M} = (S, s_0, \mathsf{Acts}, P)$ et soit un sous-ensemble d'états $F \subseteq S$. La chaîne de Markov $\mathcal{M}$ est "*globally coarse*" vis-à-vis du sous-ensemble F si et seulement si, il existe une borne minimale $\alpha > 0$ telle que pour tous les états $s \in S$, la probabilité d'atteindre fatalement F à partir de s est égale à 0 ou supérieure ou égale à α. Dans [74], les auteurs ont montré qu'une chaîne de Markov discrète $\mathcal{M}$ est "*globally coarse*" vis-à-vis du sous-ensemble F, si la probabilité d'atteindre fatalement le sous-ensemble F ou un sous-ensemble ($\widetilde{F}$) à partir duquel il n'est pas possible d'atteindre F est égale à 1 à partir de tous les états de $\mathcal{M}$.

Dans ce qui suit, nous appliquons ce résultat sur la chaîne de Markov $[\![M]\!]'$ afin de prouver que Evts_c converge presque certainement. Nous procédons à cette preuve comme suit :

(a) Nous commençons par introduire les notations qui vont être utilisées tout au long de la preuve

(b) Nous proposons une partition des états étendus T de $[\![M]\!]'$ et nous introduisons notre sous-ensemble $F \subseteq T$

(c) Nous montrons que tous les états de la partition de T satisfont la propriété de "*global coarseness* vis-à-vis de F.

(d) Nous montrons finalement que l'ensemble $\widetilde{F}$ est vide et nous concluons.

Nous détaillons dans ce qui suit chacune de ces étapes énnoncées précédemment :

(a) Nous considérons dans la suite de cette preuve les notations suivantes :

- Dans la chaîne de Markov $[\![M]\!]'$, nous partitionnons l'ensemble des actions (noms des événements) comme suit : $\mathsf{Acts} = \mathsf{Acts}_{nc} \cup \mathsf{Acts}_c$ où Acts_{nc} est le sous-ensemble d'actions correspondantes aux événements probabilistes non convergents et Acts_c est le sous-ensemble d'actions correspondantes aux événements probabilistes convergents.
- Étant donné un état étendu t et un sous-ensemble d'états $G \subseteq T$, nous dénotons par $\mathbb{P}(t \models \Diamond G)$ la probabilité de fatalement atteindre G à partir de t.
- Étant donné un prédicat $\mathcal{P}$ et un état étendu $t = (s, a)$ de $[\![M]\!]'$, nous écrivons $\mathcal{P}(t)$ pour l'évaluation de $\mathcal{P}$ dans l'état s.
- Étant donné un état étendu $t = (s, a) \in T$, nous dénotons par $\mathsf{Acts}(t)$ le sous-ensemble d'événements activables dans l'état s. De manière similaire, nous dénotons par $\mathsf{Acts}_c(t)$ l'ensemble des événements probabilistes convergents activables dans l'état s et par $\mathsf{Acts}_{nc}(t)$ l'ensemble des événements probabilistes non convergents activables dans l'état s.
- Étant donné un ensemble d'événements E et un état $t = (s, a) \in T$, nous dénotons par $W^t(E)$ la somme des poids des événements de E qui sont activables dans l'état s.
- Étant donné un état $t = (s, a) \in T$, nous dénotons par $Succ(t)$ l'ensemble des états étendus qui sont atteignables à partir de t :

$$Succ(t) = \{t' \in T | \exists e \in \mathsf{Acts}(t).P'(t, e, t') > 0\}$$

- Étant donné une exécution finie $\sigma = t_0, e_0, t_1, \ldots, t_{n-1}, e_{n-1}, t_n$ de $[\![M]\!]'$, la longueur de σ est dénoté par $L(\sigma)$, et elle est égale au nombre de transitions exécutées dans σ. Pour le cas ci-dessus de σ, $L(\sigma) = n$.

Nous utilisons également toutes les notations introduites dans la section 3.5 du chapitre 3.

(b) Nous introduisons maintenant les ensembles suivants des états étendus T :

- $T_1 = \{t = (s, a) \in T \mid \exists e \in \mathsf{Evts}_c, \exists \theta \in T_s^e, G_e(s, \theta) \wedge \forall e' \in \mathsf{Evts}_{nc}, \forall \theta \in T_s^{e'}, \neg G_{e'}(s, \theta)\}$ est l'ensemble des états étendus où seulement les événements convergents peuvent être activables.
- $T_2 = \{t = (s, a) \in T \mid \exists e \in \mathsf{Evts}_c, \exists \theta \in T_s^e, G_e(s, \theta) \wedge \exists e' \in \mathsf{Evts}_{nc}, \exists \theta \in T_s^{e'}, G_{e'}(s, \theta)\}$ est l'ensemble des états où des événements convergents et non convergents peuvent être activables.
- $T_3 = \{t = (s, a) \in T \mid \forall e \in \mathsf{Evts}_c, \forall \theta \in T_s^e, \neg G_e(s, \theta)\}$ est l'ensemble des états où aucun événement convergent n'est activable.
- $T_4 = \{t = (s, a) \in T \mid a \in \mathsf{Evts}_{nc}\}$ est l'ensemble des états atteignables après l'exécution d'un événement non convergent.

Nous pouvons constater que $T = T_1 \cup T_2 \cup T_3$ définit une partition de T. L'ensemble Evts_c converge presque certainement si nous atteignons un état dans l'ensemble $F = T_3 \cup T_4$. Nous définissons ainsi notre ensemble destination par $F = T_3 \cup T_4$. Comme en [74], nous dénotons par $\widetilde{F}$ le sous ensemble d'états de T à partir desquels il n'est pas possible d'atteindre F. Nous montrons par la suite que $\widetilde{F}$ est vide.

(c) Nous montrons maintenant que tous les états étendus des ensembles T_1, T_2 et T_3 satisfont la "*global coarseness property*" vis-à-vis de l'ensemble F, c'est-à-dire, il existe une borne minimale $\alpha > 0$ telle que pour chaque état étendu $t \in T$, la probabilité d'atteindre fatalement l'ensemble F est égale à 0 ou supérieure ou égale à α.

- Nous commençons par les états dans T_2. Soit $t_2 = (s_2, a) \in T_2$. Soit F_2 le sous ensemble d'états qui sont atteignables à partir de l'état t_2 par l'intermediaire d'un événement non convergent. Trivialement, $F_2 \subseteq T_4 \subseteq F$. Formellement,

$$F_2 = \{t' = (s', a') \in T \mid t' \in Succ(t_2) \wedge a' \in \mathsf{Acts}_{nc}\}$$

Par définition de T_2, au moins un événement convergent est activable dans t_2. Ainsi, nous avons $W^{t_2}(\mathsf{Acts}_c) > 0$. De manière similaire, au moins un événement non convergent peut être activable dans t_2 et ainsi $W^{t_2}(\mathsf{Acts}_{nc}) > 0$. Ainsi, $W^{t_2}(\mathsf{Acts}) > 0$. Nous rappelons que la probabilité d'une transition (t_2, e, t') où $e \in \mathsf{Acts}_{nc}(t_2)$ et $t' = (s', e) \in F_2$ est donnée par :

$$P'(t_2, e, t') = P(s_2, e, s') = \frac{W_e(s_2)}{W^{s_2}(\mathsf{Acts})} \times \sum_{\theta \in T^e_{s_2}} \Big[P_{T^e_{s_2}}(\theta) \times \prod_{\mathsf{x} \in Var(e)} P^e_{s_2,\theta}(\mathsf{x}, s') \Big]$$

Par définition, tous les événements non convergents e mènent le système vers des états dans F_2 indépendamment du choix probabiliste effectué dans l'action de e. Ainsi :

$$\sum_{e \in \mathsf{Acts}_{nc}(s_2), t' \in F_2} P(t_2, e, t') = \sum_{e \in \mathsf{Acts}_{nc}(s_2)} \frac{W_e(s_2)}{W^{s_2}(\mathsf{Acts})} \times 1$$

Ainsi, la probabilité de fatalement atteindre le sous-ensemble F_2 à partir de l'état t_2 est supérieure à $\frac{W^{s_2}(\mathsf{Acts}_{nc})}{W^{s_2}(\mathsf{Acts})}$. Nous montrons maintenant par contradiction qu'il existe $\alpha_2 > 0$ elle que $\forall t_2 \in T_2, \mathbb{P}(t_2 \models \Diamond F_2) \geq \alpha_2$.
Supposons le contraire, c'est-à-dire. $\forall \alpha_2 > 0, \exists t_2 \in T_2$ s.t $\mathbb{P}(t_2 \models \Diamond F_2) < \alpha_2$.
Soit α_2 telle que $(\frac{1}{\alpha_2} - 1) > BW \times card(\mathsf{Acts}_c)$. Il doit exister un état $t_2 = (s_2, a) \in T_2$ telle que $\mathbb{P}(t_2 \models \Diamond F_2) < \alpha_2$. Par le résultat présenté ci-dessus, nous savons que $\mathbb{P}(t_2 \models \Diamond F_2) \geq \frac{W^{s_2}(\mathsf{Acts}_{nc})}{W^{s_2}(\mathsf{Acts})}$. En conséquence, nous devons avoir :

$$\frac{W^{s_2}(\mathsf{Acts}_{nc})}{W^{s_2}(\mathsf{Acts})} < \alpha_2$$

Rappelons que $W^{s_2}(\mathsf{Acts}) = W^{s_2}(\mathsf{Acts}_{nc}) + W^{s_2}(\mathsf{Acts}_c)$. Ainsi,

$$\frac{W^{s_2}(\mathsf{Acts})}{W^{s_2}(\mathsf{Acts}_{nc})} = 1 + \frac{W^{s_2}(\mathsf{Acts}_c)}{W^{s_2}(\mathsf{Acts}_{nc})} > \frac{1}{\alpha_2}$$

En conséquence,

$$W^{s_2}(\mathsf{Acts}_c) > W^{s_2}(\mathsf{Acts}_{nc}) \cdot (\frac{1}{\alpha_2} - 1)$$

Par définition de T_2, nous avons $W^{s_2}(\mathsf{Acts}_{nc}) \geq 1$, ainsi

$$W^{s_2}(\mathsf{Acts}_c) > (\frac{1}{\alpha_2} - 1)$$

Finalement, par définition de α_2, nous avons $W^{s_2}(\mathsf{Acts}_c) > BW \times card(\mathsf{Acts}_c)$, ce qui est clairement en contradiction avec l'obligation de preuve event/wght/BOUND.
Nous pouvons ainsi conclure qu'il existe $\alpha_2 > 0$ telle que $\forall t_2 \in T_2, \mathbb{P}(t_2 \models \Diamond F_2) \geq \alpha_2$.

- Nous passons maintenant aux états qui sont dans T_1 : nous montrons qu'il existe α_1 tel que pour tous les états étendus $t_1 \in T_1, \mathbb{P}(t_1 \models \Diamond F) \geq \alpha_1$.
Nous rappelons que la fonction de probabilité de $[\![M]\!]'$ est donnée comme suit : Pour tout $t_1 = (s_1, a) \in T_1$, $e \in \mathsf{Acts}$, et $t' = (s', a) \in T$, nous avons

$$P(t_1, e, t') = P(s_1, e, s') = \frac{W_e(s_1)}{W^{s_1}(\mathsf{Acts})} \times \sum_{\theta \in T^e_{s_1}} \Big[P_{T^e_{s_1}}(\theta) \times \prod_{\mathsf{x} \in Var(e)} P^e_{s_1,\theta}(\mathsf{x}, s') \Big]$$

Puisque $t_1 \in T_1$, cette expression peut être supérieure à zéro si e est un événement convergent. Dans ce cas, l'obligation de preuve event/wght/BOUND assure que $W^{s_1}(\mathsf{Acts}) \leq BW \cdot card(\mathsf{Acts}_c)$. Ainsi, pour tous les événements convergents activables dans t_1, nous avons $\frac{W_e(s_1)}{W^{s_1}(\mathsf{Acts})} \geq \frac{1}{BW \cdot card(\mathsf{Acts}_c)}$.

De plus, l'obligation de preuve event/param/BOUND assure que le nombre des valeurs des paramètres qui satisfont la garde de l'événement e dans l'état s_1 est majoré par BP. En conséquence,

$$\sum_{\theta \in T^e_{s_1}} \Big[P_{T^e_{s_1}}(\theta) \times \prod_{\mathsf{x} \in Var(e)} P^e_{s_1,\theta}(\mathsf{x}, s') \Big] \geq \frac{1}{BP} \times \sum_{\theta \in T^e_{s_1}} \Big[\prod_{\mathsf{x} \in Var(e)} P^e_{s_1,\theta}(\mathsf{x}, s') \Big]$$

Finalement, puisque les valeurs de probabilité à l'intérieur des substitutions probabilistes ($P^e_{\mathsf{x}}(E)$) sont constantes et finies, il existe une valeur minimale $\beta > 0$ telle que pour tout $t_1 = (s_1, a) \in T_1, e \in Acts_c$, et $t' = (s', e) \in T$, telle que $P(t_1, e, t') > 0$, nous avons

$$\sum_{\theta \in T^e_{s_1}} \Big[\prod_{\mathsf{x} \in Var(e)} P^e_{s_1,\theta}(\mathsf{x}, s') \Big] \geq \beta$$

En conséquence, il existe une valeur minimale $\gamma > 0$ telle que $P'(t_1, e, t') \geq \gamma$ pour tout $t_1 \in T_1, e \in Acts_c$, et $t' \in T$ telle que $P'(t_1, e, t') > 0$.

Soit $t_0 = (s_0, a_O) \in T_1$ un état étendu. Par définition de T_1 et à cause des obligations de preuve event/pBOUND, event/var/pNAT et machine/pVar, la valeur du variant dans t_0 est comprise entre 0 et U et il doit exister une transition qui mène le système vers un état étendu $t_1 = (s_1, a_1)$ telle que $V(t_1) < V(t_0)$. Nécessairement, nous avons $t_1 \in T_1$ ou $t_1 \in T_2$ ou $t_1 \in T_3$, ainsi, il doit exister une exécution finie $\sigma = t_0, e_0, t_1, \ldots, t_{n-1}, t_{n-1}, t_n$ avec $t_n \in T_2 \cup T_3$ et $\forall i < n$, $t_i \in T_1$ et $L(\sigma) \leq U + 1$.

Si $t_n \in T_3 \subseteq F$, alors $\mathbb{P}(t_0 \models \Diamond F) \geq \gamma^{U+1}$. Autrement, nous avons $t_n \in T_2$ et $\mathbb{P}(t_n \models \Diamond F) \geq \alpha_2$, ainsi, $\mathbb{P}(t_0 \models \Diamond F) \geq \alpha_2 \cdot \gamma^{U+1}$.

En conséquence, étant donné $\alpha_2 \leq 1$, nous avons $\gamma^{U+1} \geq \alpha_2 \cdot \gamma^{U+1}$ et il existe $\alpha_1 = \alpha_2 \cdot \gamma^{U+1} > 0$ tel que pour tous les états étendus $t_1 \in T_1, \mathbb{P}(t_1 \models \Diamond F) \geq \alpha_1$.

- Finalement, étant donné que $T_3 \subseteq F$, nous avons $\mathbb{P}(t_3 \models \Diamond F) = 1$ pour tous les états étendus $t_3 \in T_3$.

Ainsi, nous concluons que $[\![M]\!]'$ est "*globally coarse*" vis-à-vis l'ensemble d'états F. En conséquence, $\forall t \in T, \mathbb{P}(t \models \Diamond F \vee \Diamond \widetilde{F}) = 1$.

(d) Nous avons montré précédemment que pour tous les états étendus dans T_1, T_2 ou T_3, nous avons $\mathbb{P}(t \models \Diamond F) > 0$. Puisque $T = T_1 \cup T_2 \cup T_3$, $\widetilde{F}$ est donc nécessairement vide.

Puisque $[\![M]\!]'$ est "*globally coarse*" vis-à-vis de l'ensemble F et que l'ensemble $\widetilde{F}$ est vide, nous avons que pour tous les états étendus $t \in T$, la probabilité d'atteindre fatalement l'ensemble destination F est 1. En conséquence, la probabilité d'atteindre fatalement un état de blocage ou un état de la forme $t = (s, e)$ où e est un événement non convergent est égale 1 à partir de tous les états (étendus) où des événements convergent sont activables, ce qui conclut notre preuve □.

Cas particulier de bornes maximales variables Nous avons proposé un ensemble d'obligations de preuve suffisant pour prouver qu'un ensemble d'événements probabilistes converge presque certainement. Dans cette proposition, nous avons considéré que les bornes maximales pour la valeur du variant U, pour la valeur du poids d'un événement convergent BW et pour la valeur du nombre de paramètres d'un événement convergent BP étaient toutes des constantes. Nous avons opté pour ce choix principalement pour des raisons de simplicité. Cependant, tout le raisonnement présenté précédemment peut être étendu au cas où ces bornes sont des fonctions dépendant des variables de la machine comme dans [54], c'est à dire une fonction BW($\bar{v}$) pour les valeurs des poids, une fonction U($\bar{v}$) pour la valeur du variant et une fonction BP($\bar{v}$) pour le nombre de valeurs de paramètres. Dans ce cas, il est cependant nécessaire, que chaque événement probabiliste convergent ne fasse pas décroître la valeur de chacune de ces bornes.

5.3.3 Étude de cas : convergence presque certaine dans le protocole P2P

Revenons à notre étude de cas du protocole pair à pair. Dans la section 2.5, nous avons présenté la machine $P2P_3$ qui donne une description détaillée de ce protocole. Cette machine est obtenue après l'ajout de l'événement FailureDL dans la machine $P2P_2$. Comme expliqué dans la section 2.5, l'événement FailureDL ajouté par raffinement dans $P2P_2$ ne converge pas car son action ne décroit pas le variant, et ainsi cette étape de raffinement n'est pas correcte. De plus, prouver la convergence de l'événement FailureDL permet non seulement de prouver la correction du raffinement mais aussi de prouver la terminaison du protocole. En effet, puisque l'événement Start1DL représente le déclenchement du téléchargement d'un bloc par un client, l'événement Finish1DL représente la finalisation de téléchargement d'un bloc par un client et l'événement FailureDL représente l'échec de téléchargement d'un bloc par un client, nous concluons que la démonstration de la convergence de ces événements permet de montrer la terminaison du protocole. Si l'un de ces événements ne converge pas alors le protocole ne termine pas.

Nous présentons dans la figure 5.8 plusieurs processus de développement B événementiel qui pourraient permettre la résolution de ce problème de convergence. La première possibilité consiste à ajouter une version probabiliste de FailureDL dans la machine $P2P_2$ obtenant ainsi la machine $P2P_m$ et d'étudier la convergence presque certaine de cet événement. Cependant, nous montrons que cet événement ne converge pas presque certainement. La deuxième possibilité consiste à partir de la machine $P2P_1$, de la raffiner par une machine $P2P_1$' dans laquelle nous ajoutons la variable DBin. Nous raffinons par la suite la machine $P2P_1$' par ajout simultané des événements Start1DL, Finish1DL et FailureDL dans leur version probabiliste. La machine obtenue est $P2P_1$'', et nous montrons dans ce cas que cet ensemble d'événements converge presque certainement et que le protocole termine. Cependant, dans cette machine, le téléchargement progressif des blocs par les clients peut s'arrêter à tout instant et l'événement AllDL finit le téléchargement en une seule opération. Nous raffinons par la suite l'événement AllDl par l'événement DLFinished comme présenté dans la section 2.5.3. La machine obtenue est alors $P2P_M$.

Notons que nous aurions pu raffiner directement la machine $P2P_1$ par la machine $P2P_1$'' sans passer par la machine intermédiaire $P2P_1$', mais nous avons opté pour ce choix car notre proposition de raffinement probabiliste dans le B événementiel probabiliste ne traite pas actuellement l'ajout de nouveaux événements probabilistes et le raffinement de données en même temps. De même, nous ne pouvons pas en même temps ajouter simultanément les événements Start1DL, Finish1DL et FailureDL dans leur version probabiliste et raffiner AllDL par DLFinished car nous avons seulement étudié l'ajout de nouveaux événements probabilistes sans raffinement d'autres événements. L'ajout de nouveaux événements probabilistes avec le raffinement de données ou le raffinement d'autres événements est une perspective de ce travail de raffinement.

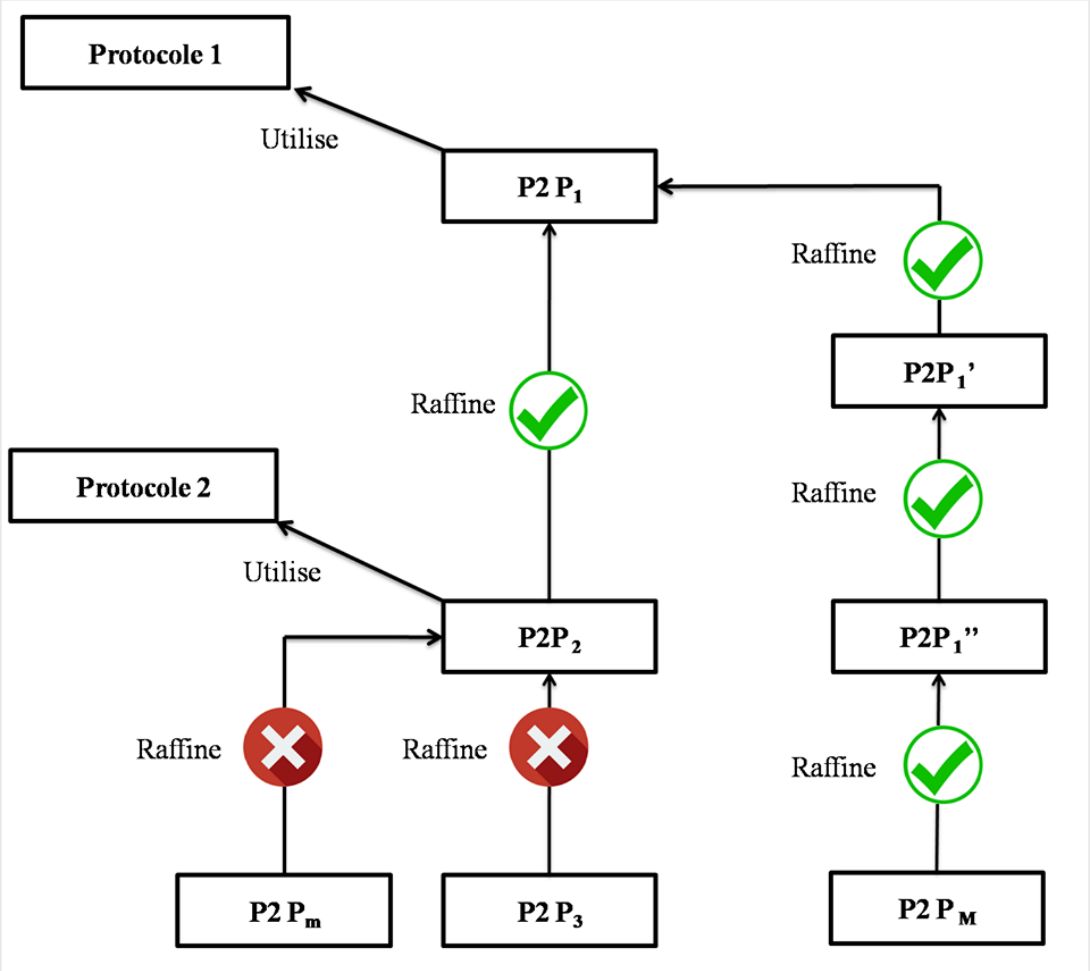

FIGURE 5.8 – Processus de développement du protocole P2P

Première possibilité de raffinement Nous ajoutons une version probabiliste de FailureDL dans la machine $P2P_2$. Nous obtenons la machine $P2P_m$ présentée dans la figure 5.9. L'événement FailureDL est annoté maintenant par le poids

$$N \times K - \text{card}(\{c{\mapsto}b \mid c \in 1..N \wedge b \in 1..K \wedge \text{DBin}(c{\mapsto}b) = \text{finished}\})$$

Nous remplaçons aussi la substitution non-déterminisme énumérée $\text{DBin}(c{\mapsto}b) :\in \{\text{empty}, \text{incoming}\}$ par une substitution probabiliste énumérée $\text{DBin}(c{\mapsto}b) := \text{empty}\ @4/10 \oplus \text{incoming}\ @6/10$.

Nous devons montrer que cet événement converge presque certainement. Pour cela, nous montrons la satisfaction des obligations de preuve présentées dans la section précédente :

- L'obligation de preuve evt/var/pNAT est bien respectée, le variant est un entier strictement positif ;
- L'obligation de preuve evt/pBOUND est bien respectée, nous considérons la constante $2 \times N \times K$ comme une borne maximale du variant ;
- L'obligation de preuve evt/wght/BOUND est bien respectée, le poids de FailureDL est majoré par l'expression $N \times K + 1$;
- L'obligation de preuve evt/param/pBOUND est bien respectée, le nombre de valeurs possibles des paramètres de FailureDL est majoré par la constante $N \times K$;
- L'obligation de preuve machine/pVAR n'est pas respectée. En effet, le seul événement probabiliste dans cette machine est FailureDL et dans toutes les valuations des variables où au moins

MACHINE
P2P$_m$
REFINES
P2P$_2$
VARIABLES
DB
DBin
INVARIANTS
DB $\in$ 1..N $\times$ 1..K $\rightarrow$ {empty,finished} $\wedge$
DBin $\in$ 1..N $\times$ 1..K $\rightarrow$ {empty,incoming,finished} $\wedge$
$\forall$ c . (c $\in$ 1..N $\Rightarrow$ card({b $\mid$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming}) $\leq$ 1)
VARIANT
2 $\times$ N $\times$ K $-$ 2 $\times$ card({c$\mapsto$b $\mid$ c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = finished})
$-$ card({c$\mapsto$b $\mid$ c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming})
EVENTS
Init $\widehat{=}$
begin
DB := (1..N $\times$ 1..K) $\times$ {empty} ||
DBin := (1..N $\times$ 1..K) $\times$ {empty}
end
DLFinished $\widehat{=}$
when
DB = (1..N $\times$ 1..K) $\times$ {empty} $\wedge$
DBin = (1..N $\times$ 1..K) $\times$ {finished}
then
DB := DBin
end
Start1DL $\widehat{=}$
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = empty $\wedge$
card({k $\mid$ k $\in$ 1..K $\wedge$ DBin(c$\mapsto$k) = incoming})=0
then
DBin(c$\mapsto$b) := incoming
end
Finish1DL $\widehat{=}$
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming
then
DBin(c$\mapsto$b) := finished
end
FailureDL $\widehat{=}$
weight
N $\times$ K $-$ card({c$\mapsto$b $\mid$ c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = finished})
any c, b **where**
c $\in$ 1..N $\wedge$ b $\in$ 1..K $\wedge$ DBin(c$\mapsto$b) = incoming
then
DBin(c$\mapsto$b) := empty @ 4/10 $\oplus$ incoming @ 6/10
end
END

FIGURE 5.9 – Une machine B événementiel mixte du protocole P2P

un client c a commencé le téléchargement d'un bloc b, l'action de cet événement ne diminue pas la valeur du variant.

Nous concluons ainsi que ce raffinement n'est pas correct.

Deuxième possibilité de raffinement La deuxième possibilité de raffinement consiste en trois étapes de raffinement. La première étape consiste à raffiner la machine $P2P_1$ en ajoutant uniquement la variable DBin (cette variable est expliquée de la même manière que dans la section 2.5.3, l'événement d'intialisation est raffiné pour prendre en compte l'initialisation de la variable DBin). La machine résultante est $P2P_1$', elle est présentée dans la figure 5.10.

MACHINE
$P2P_1$'
REFINES
$P2P_1$
SEES
protocole2
VARIABLES
DB
DBin
INVARIANTS
$DB \in 1..N \times 1..K \rightarrow \{empty,finished\}$
$DBin \in 1..N \times 1..K \rightarrow \{empty,incoming,finished\} \wedge$
$\forall c \cdot (c \in 1..N$
$\Rightarrow card(\{b \mid b \in 1..K \wedge DBin(c \mapsto b) = incoming\}) \leq 1)$

EVENTS
Init $\widehat{=}$
begin
$DB := (1..N \times 1..K) \times \{empty\} \parallel$
$DBin := (1..N \times 1..K) \times \{empty\}$
end

AllDL $\widehat{=}$
any magicDB **where**
$DB = (1..N \times 1..K) \times \{empty\} \wedge$
$magicDB \in 1..N \times 1..K \rightarrow \{empty,finished\} \wedge$
$magicDB = (1..N \times 1..K) \times \{finished\}$
then
$DB := magicDB$
end

END

FIGURE 5.10 – Machine $P2P_1$'

La deuxième étape consiste à ajouter simultanément les événements Start1DL, Finish1DL et FailureDL dans leur version probabiliste. À ce stade de raffinement, nous supposons que le téléchargement progressif des blocs par les clients peut s'arrêter à un certain moment et que l'événement AllDL finit le téléchargement de tous les blocs en une seule opération. La machine résultante est $P2P_1$'', elle est présentée dans la figure 5.11.

Le choix des poids des événements Start1DL, Finish1DL et FailureDL ainsi que les valeurs de probabilité dans la substitution de FailureDL est expliqué de la même manière que dans la section 5.2.3. Puisque nous introduisons de nouveaux événements probabilistes, nous devons prouver leur convergence presque certaine. Cette machine possède le même variant que la machine $P2P_2$. Pour prouver que l'ensemble des événements {Start1DL, Finish1DL, FailureDL} converge presque

MACHINE
$P2P_1$''
REFINES
$P2P_1$'
VARIABLES
DB
DBin
INVARIANTS
$DB \in 1..N \times 1..K \rightarrow \{empty,finished\} \wedge DBin \in 1..N \times 1..K \rightarrow \{empty,incoming,finished\} \wedge$
$\forall c \cdot (c \in 1..N \Rightarrow card(\{b \mid b \in 1..K \wedge DBin(c \mapsto b) = incoming\}) \leq 1)$
VARIANT
$2 \times N \times K - 2 \times card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = finished\})$
$- card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming\})$
EVENTS
Init $\widehat{=}$
begin
$DB := (1..N \times 1..K) \times \{empty\} \parallel$
$DBin := (1..N \times 1..K) \times \{empty\}$
end
AllDL $\widehat{=}$
any magicDB **where**
$DB = (1..N \times 1..K) \times \{empty\} \wedge$
$magicDB \in 1..N \times 1..K \rightarrow \{empty,finished\} \wedge$
$magicDB = (1..N \times 1..K) \times \{finished\}$
then
$DB := magicDB$
end
Start1DL $\widehat{=}$
weight
$N \times K - card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming\})$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = empty \wedge card(\{k \mid k \in 1..K \wedge DBin(c \mapsto k) = incoming\})=0$
then
$DBin(c \mapsto b) := incoming$
end
Finish1DL $\widehat{=}$
weight
$card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = finished\}) +1$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming$
then
$DBin(c \mapsto b) := finished$
end
FailureDL $\widehat{=}$
weight
$N \times K - card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = finished\})$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming$
then
$DBin(c \mapsto b) := empty \ @ \ 4/10 \oplus incoming \ @ \ 6/10$
end

FIGURE 5.11 – Machine $P2P_1$''

certainement (converge avec une probabilité 1), nous devons montrer que ces événements satisfont les obligations de preuve présentées dans la section précédente :

- L'obligation de preuve evt/var/pNAT est bien respectée, le variant est un entier strictement positif ;
- L'obligation de preuve evt/pBOUND est bien respectée, nous considérons la constante $2 \times N \times K$ comme une borne maximale du variant ;
- L'obligation de preuve evt/wght/BOUND est bien respectée, les poids des événements Start1DL, Finish1DL, FailureDL sont majorés par l'expression $N \times K + 1$;
- L'obligation de preuve evt/param/pBOUND est bien respectée, le nombre de valeurs possibles des paramètres des événements Start1DL, Finish1DL, FailureDL est majoré par la constante $N \times K$;
- L'obligation de preuve machine/pVAR est bien respectée : dans toutes les valuations des variables où au moins un client c a commencé le téléchargement d'un bloc b, l'événement Finish1DL décroit la valeur du variant avec une valeur de probabilité strictement positive.

Puisque l'ensemble des événements {Start1DL, Finish1DL, FailureDL} satisfait les obligations de preuve nécessaires, le Thèorème 1 garantit que cet ensemble converge presque certainement. Cette preuve de convergence garantit la correction du raffinement. De même, comme expliqué précédemment, elle garantit la terminaison du protocole, c'est-à-dire, on arrive avec une probabilité 1 à un état où tous les clients ont réussi à télécharger tous les blocs.

La dernière étape de raffinement consiste à raffiner l'événement AllDL par l'événement DLFinished comme présenté dans le premier raffinement décrit dans la machine $P2P_2$ de la figure 2.9 (l'événement DLFinished est expliqué de la même manière que dans la section 2.5.3). À ce stade de raffinement, le téléchargement de blocs par tous les clients se fait d'une manière progressive, et il n'y a plus de possibilité de téléchargement total de tous les blocs par tous les clients en une seule opération. La machine résultante est $P2P_M$, elle est présentée dans la figure 5.12. Cette machine garantit la terminaison du protocole puisqu'elle résulte d'un raffinement correct de la machine $P2P_1$'' dans laquelle la terminaison du protocole est déjà garantie.

5.4 Généralisation de l'introduction d'événements par raffinement

Nous traitons maintenant le cas général d'introduction d'événements de n'importe quel type (standard ou probabilistes) dans une machine B événementiel de n'importe quel type (standard, probabiliste ou mixte). Nous montrons que les résultats présentés précédemment peuvent être facilement généralisés à un cas plus général lorsque nous combinons l'ajout de nouveaux événements probabilistes et standard au sein d'une machine B événementiel. Dans cette section, nous traitons trois cas d'ajout de nouveaux événements :

1. L'ajout de nouveaux événements probabilistes dans une machine B événementiel standard ou mixte ;
2. L'ajout de nouveaux événements standard dans une machine B événementiel probabiliste ou mixte ;
3. L'ajout simultané d'événements probabilistes et standard dans une machine de n'importe quel type.

Nous détaillons ces trois cas dans ce qui suit.

MACHINE
$P2P_M$
REFINES
$P2P_1$''
VARIABLES
DB
DBin
INVARIANTS
$DB \in 1..N \times 1..K \to \{empty,finished\} \wedge DBin \in 1..N \times 1..K \to \{empty,incoming,finished\} \wedge$
$\forall c \cdot (c \in 1..N \Rightarrow card(\{b \mid b \in 1..K \wedge DBin(c \mapsto b) = incoming\}) \leq 1)$
VARIANT
$2 \times N \times K - 2 \times card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = finished\})$
$- card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming\})$
EVENTS
Init $\widehat{=}$
begin
$DB := (1..N \times 1..K) \times \{empty\} \parallel$
$DBin := (1..N \times 1..K) \times \{empty\}$
end
DLFinished $\widehat{=}$
refines AllDL
when
$DB = (1..N \times 1..K) \times \{empty\} \wedge DBin = (1..N \times 1..K) \times \{finished\}$
then
$DB := DBin$
end
Start1DL $\widehat{=}$
weight
$N \times K - card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming\})$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = empty \wedge card(\{k \mid k \in 1..K \wedge DBin(c \mapsto k) = incoming\}) = 0$
then
$DBin(c \mapsto b) := incoming$
end
Finish1DL $\widehat{=}$
weight
$card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = finished\}) + 1$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming$
then
$DBin(c \mapsto b) := finished$
end
FailureDL $\widehat{=}$
weight
$N \times K - card(\{c \mapsto b \mid c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = finished\})$
any c, b **where**
$c \in 1..N \wedge b \in 1..K \wedge DBin(c \mapsto b) = incoming$
then
$DBin(c \mapsto b) := empty \ @ \ 4/10 \oplus incoming \ @ \ 6/10$
end
END

FIGURE 5.12 – Machine $P2P_M$

5.4.1 Ajout de nouveaux événements probabilistes au sein d'une machine B événementiel standard ou mixte

Indépendamment du type de la machine initiale, le résultat de l'introduction de nouveaux événements probabilistes va être une machine B événementiel mixte. Les résultats présentés dans la section 5.3.2 peuvent être adaptés facilement au cas des machines B événementiel standards ou mixtes. En introduisant de nouveaux événements probabilistes, nous avons uniquement à prouver la convergence presque certaine de ces nouveaux événements. Ainsi, les obligations de preuve nécessaires sont identiques à celles présentées dans la section 5.3.1.

Proposition 2

Soit M=($\bar{v}$,I($\bar{v}$),V($\bar{v}$),MEvts,Init) une machine B événementiel mixte. MEvts= MEvts$_{pc}$ ∪ MEvts$_{pnc}$ ∪ MEvts$_{nd}$ est la partition de l'ensemble des événements de M en l'ensemble des nouveaux événements probabilistes MEvts$_{pc}$, l'ensemble des anciens événements probabilistes MEvts$_{pnc}$ et l'ensemble des événements non-déterministes MEvts$_{nd}$. Si l'ensemble des événements MEvts$_{pc}$ satisfait les obligations de preuve (machine/pVar), (evt/wght/BOUND), (evt/param/BOUND) et (evt/var/NAT) présentées dans la section 5.3.1, alors MEvts$_{pc}$ converge presque certainement (avec probabilité 1, indépendemment des choix non-déterministes).

Squelette de la preuve Puisque uniquement certains événements probabilistes doivent être convergents, les choix non-déterministes n'ont pas d'impact sur la propriété de convergence presque certaine. Nous pouvons ainsi adapter facilement la preuve du Théorème 1 à la sémantique sous forme d'automate probabiliste en utilisant des "*schedulers*" [28]. Le rôle des schedulers est trivial : indépendamment du *scheduler* choisi, la probabilité de fatalement exécuter un événement non convergent est 1 □.

5.4.2 Ajout d'événements standard dans une machine B événementiel mixte ou probabiliste

Dans la section précédente, nous avons expliqué comment introduire des événements probabilistes dans une machine B événementiel standard ou mixte, nous considérons maintenant l'opération inverse, c'est-à-dire l'introduction de nouveaux événements standard dans une machine B événementiel probabiliste ou mixte. Comme précédemment, puisque uniquement les nouveaux événements standard doivent converger, nous constatons évidemment que les obligations de preuve de convergence standard présentées dans la section 2.4.3 sont suffisantes pour prouver la convergence de nouvaux événements standard.

5.4.3 Ajout simultané d'événements standard et probabilistes

Nous traitons maintenant l'ajout simultané d'événements standards et probabilistes dans une seule étape de raffinement. Indépendamment du type de la machine dans laquelle nous ajoutons ces événements, la machine résultante va être une machine B événementiel mixte. Nous devons assurer à la fois la convergence des événements probabilistes et des événements standards.

Rappelons que prouver la convergence d'un ensemble d'événements standards revient à prouver que dans toutes les valuations des variables où au moins un événement convergent est activable, chaque événement convergent activable décroit le variant. Dans notre proposition, nous avons montré qu'un ensemble d'événements probabilistes converge lorsque dans toutes les valuations des variables où des événements probabilistes sont activables, au moins l'un de ces événements décroit le variant.

Lorsqu'il s'agit de la preuve simultanée de convergence d'événements probabilistes et standards, nous proposons de mélanger les deux approches. Nous imposons que dans toutes les valuations où des événements convergents (peu importe leur type) sont activables, au moins l'un des événements probabilistes convergents doit décroître le variant et que tous les événements standards convergents doivent décroître le variant. Ainsi, il est nécessaire de prouver les obligations de preuve de convergence standard pour les événements standard et les obligations de preuve de la convergence presque certaine pour les événements probabilistes. La seule modification concerne la borne maximale pour le variant : tous les événements convergents (standard ou probabilistes) doivent respecter la condition sur la borne maximale pour le variant introduite dans la section 5.3.1. Ainsi, nous avons une nouvelle obligation de preuve pour les événements standard, nous la présentons dans ce qui suit.

Variant majoré pour les événements standards Cette obligation de preuve assure que les événements standard convergents ne sont activables que lorsque le variant est majoré par la borne maximale U.

evt/ndpBOUND

$$\mathsf{I}(\bar{v}) \wedge$$
$$\mathsf{G}_i(\bar{t}, \bar{v}) \wedge$$
$$\vdash$$
$$\mathsf{V}(\bar{v}) \leq \mathsf{U}$$

Théorème 2

Soit M=($\bar{v}$,I($\bar{v}$),V($\bar{v}$), Init , MEvts) une machine B événementiel mixte (totale). Soit MEvts= MEvts_{pc} ∪ MEvts_{pnc} ∪ MEvts_{ndc} ∪ MEvts_{ndnc} la partition de l'ensemble des événements MEvts vers l'ensemble des nouveaux événements probabilistes MEvts_{pc}, l'ensemble des autres événements probabilistes MEvts_{pnc}, l'ensemble des nouveaux événements standards MEvts_{ndc} et l'ensemble des autres événements standards MEvts_{ndnc}. Si tous les événements de MEvts_{pc} ∪ MEvts_{ndc} satisfont l'obligation de preuve (evt/var/NAT) présentée dans la section 2.4.3, si tous les événements de MEvts_{ndc} satisfont les obligations de preuve (evt/var) (à partir de la section 2.4.3) et (evt/ndpBOUND) présentée ci-dessus, si tous les événements de MEvts_{pc} satisfont les obligations de preuve (evt/wght/BOUND), (evt/pBOUND) et (evt/var/pNAT) présentées dans la section 5.3.1, alors l'ensemble des événements MEvts_{pc} ∪ MEvts_{ndc} converge presque certainement dans M, c'est-à-dire converge avec une probabilité 1 dans le pire des cas.

Squelette de la preuve Comme c'est le cas de la proposition 1, la preuve du Théorème 1 peut être facilement adapté à ce cas là. Nous utilisons aussi ici des *schedulers*. L'idée d'abord est de fixer un "*scheduler*" arbitraire. Ensuite, nous montrons qu'avec ce *scheduler*, la probabilité de fatalement exécuter un événement non convergent est de 1. Puisque tous les choix non-détemernistes mènent le système à

- soit un événement standard convergent (qui décroit le variant avec une valeur de probabilité 1),
- soit un événement non convergent (qui finit ainsi cette preuve),

- soit vers une transition probabiliste combinée (qui décroit le variant avec une valeur de probabilité strictement positive ou elle n'est pas convergente).

Alors, la conclusion est facilement obtenue. Tous les choix non-déterministes ménent le système à :

(1) soit un événement non-détemerniste convergent qui décroit la valeur du variant ;

(2) soit la transition probabiliste combinée qui soit elle décroit le variant avec une probabilité strictement positive, soit elle n'est pas convergente ;

(3) un événement non-convergent.

Dans les cas (1) et (2), le variant va décroitre, pour le cas (3), notre preuve se termine. Ainsi, la conclusion est facilement obtenue. □.

6

Études de cas

Afin d'illustrer les éléments de notre extension à B événementiel présentés tout au long de ce manuscrit, nous présentons dans ce chapitre deux études de cas : un système de train d'atterrissage et un système de freinage. La première étude de cas nous permet d'illustrer le processus de probabilisation présenté dans la section 5.2. L'étude de cas du système de freinage nous permet d'illustrer l'ajout de nouveaux événements probabilistes par raffinement présenté dans la section 5.3.

6.1 Système d'atterrissage d'un avion

Ce cas d'étude représente le système d'atterrissage d'un avion. Ce système s'occupe de la gestion du train d'atterrissage, c'est-à-dire, du contrôle des trappes et des trains correspondants. Dans cette section, nous traitons une version simplifiée de ce système dont une description détaillée est fournie dans [35]. Nous présentons cette étude de cas afin de montrer un exemple d'application réel du processus de *probabilisation* présenté dans la section 5.2. La figure 6.1 représente le principe de fonctionnement du train d'atterrissage. Le pilote peut commander le système de train d'atterrissage par l'intermédiaire d'un levier. Le comportement du train d'atterrissage peut se diviser en deux séquences : une séquence d'extension et une séquence de rétraction. En fonction de son besoin, le pilote peut commander l'une de ces séquences.

Séquence d'extension :

1. Ouverture des trappes,
2. Extension des trains d'atterrissage,
3. Fermeture des trappes.

Séquence de rétraction :

1. Ouverture des trappes,
2. Rétraction des trains d'atterrissage,
3. Fermeture des trappes.

Lorsque l'avion est en état de vol, le pilote peut commander l'exécution de l'une des séquences. Le système du train d'atterrissage doit satisfaire certaines exigences. En effet, le pilote ne peut pas commander le levier plus qu'un certain nombre de fois sans que l'une des séquences d'extension ou de rétraction soit exécutée. De plus, plus le pilote commande le levier, plus la chance de le commander encore une fois diminue et la chance d'exécuter l'une des séquences augmente. Dans certains cas, des pannes peuvent affecter le comportement des trappes et des trains d'atterrissage : il est

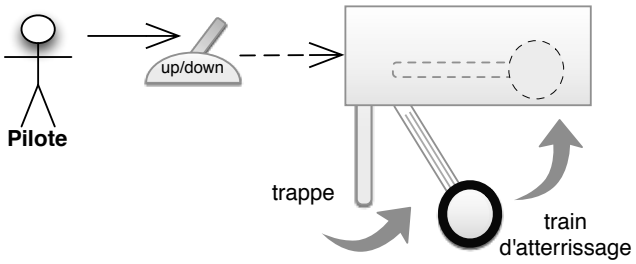

FIGURE 6.1 – Principe du fonctionnement du train d'atterrissage

possible que les trappes ne s'ouvrent pas ou que les trains ne s'étendent pas suite à la commande du levier.

Nous donnons dans ce qui suit une description précise de ces exigences :

(E1) À partir d'un certain nombre de commandes du levier par le pilote, l'une des séquences doit être obligatoirement exécutée ;

(E2) Plus le pilote commande le levier, plus la chance d'exécuter l'une des séquences d'extension ou de rétraction augmente ;

(E3) Les trappes ou les trains peuvent tomber en panne avec un pourcentage de 10%.

6.2 Stratégie de développement du train d'atterrissage en B événementiel

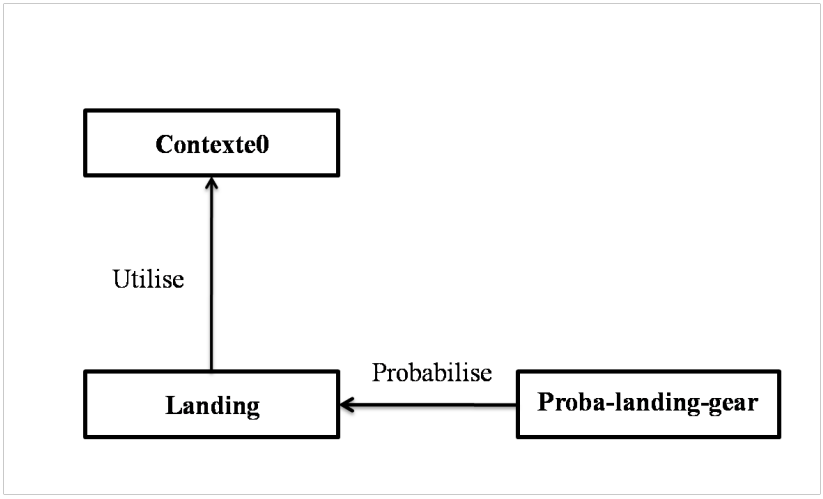

FIGURE 6.2 – Stratégie de développement du système du train d'atterrissage en B événementiel

Nous résumons dans la figure 6.2 la stratégie de développement adoptée pour la spécification de cette étude de cas. La machine Landing présente une description du système en B événementiel. La machine Proba−landing−gear est le résultat de l'application du processus de probabilisation sur la machine Landing. Les deux machines utilisent le même contexte contexte0.

6.2.1 La première machine

CONTEXT
contexte0
CONSTANTS
up
down
open
closed
extended
retracted
FCMD
SETS
HandleState
DoorState
GearState
AXIOMS
partition (HandleState,{up},{down})
partition (DoorState,{open},{closed})
partition (GearState,{extended},{ retracted }
END

FIGURE 6.3 – Contexte initial

Le système décrit le comportement du train d'atterrissage d'un avion. Le levier de commande peut être en position haute ou basse. Pour la description de l'état de ce levier, nous créons deux constantes up et down. La trappe du train d'atterrissage peut être ouverte ou fermée. Pour la description de son état, nous créons deux constantes open et closed. De même, pour la description de l'état du train d'atterrissage, nous créons deux constantes extended et retracted.

Tous ces éléments sont définis dans le contexte contexte0 présenté dans la figure 6.3.

La machine présentée dans la figure 6.4 permet de spécifier les éléments de base du comportement du train d'atterrissage. Dans cette machine, l'état du système est décrit par quatre variables : handle, door, gear et cmd.

- handle modélise l'état du levier du pilote,
- door modélise l'état de la trappe du train d'atterrissage,
- gear modélise l'état du train d'atterrissage lui même,
- cmd représente le nombre de fois où le levier a été commandé par le pilote avant que l'une des séquences d'extension ou de rétraction soit exécutée.

Initialement, nous considérons l'avion en état de vol. Le pilote n'a pas encore manipulé le levier, les trappes sont fermées et le train est rétracté. La machine contient cinq événements :

- L'événement PilotCmd modélise la commande du levier par le pilote. Nous remarquons que l'action de cet événement augmente la valeur de cmd représentant le nombre de fois où le levier a été commandé par le pilote sans exécution d'aucune séquence ;
- L'événement extend modélise l'extension du train d'atterrissage. Cet événement n'est activable qu'après que le pilote commande l'exécution de la séquence d'extension par son levier (handle= down). Nous utilisons une substitution non-déterministe énumérée gear :∈ {extended,

MACHINE
Landing
INVARIANTS
handle $\in$ HandleState
door $\in$ DoorState
gear $\in$ GearState
cmd $\in \mathbb{N}$
EVENTS
Init $\widehat{=}$
handle := up
door := closed
gear := retracted
cmd := 0
PilotCmd $\widehat{=}$
any
cc
where
cc $\in$ {up,down} $\wedge$ cmd $\leq$ FCMD
then
handle := cc
cmd := cmd+1
end
extend $\widehat{=}$
when
handle = down $\wedge$ door = open $\wedge$ gear = retracted
then
gear :$\in$ {extended, retracted}
cmd := 0
end
retract $\widehat{=}$
when
handle = up $\wedge$ door = open $\wedge$ gear = extended
then
gear :$\in$ {extended, retracted}
end
open $\widehat{=}$
when
door = closed $\wedge$ ((handle = down $\wedge$ gear = retracted)
$\vee$ (handle = up $\wedge$ gear = extended))
then
door :$\in$ {open,closed}
end
close $\widehat{=}$
when
door = open $\wedge$ ((handle = down $\wedge$ gear = extended)
$\vee$ (handle = up $\wedge$ gear = retracted))
then
door :$\in$ {closed,open}
end
END

FIGURE 6.4 – Première machine

retracted } pour exprimer le fait que la variable gear peut passer à extended, représentant ainsi le comportement attendu, ou rester à retracted , représentant ainsi le cas de panne, c'est-à-dire que le train ne réagit pas à la commande ;

- L'événement retract modélise la rétraction du train d'atterrissage. De la même manière que précédemment, nous utilisons une substitution non-déterministe énumérée gear $:\in$ {extended, retracted } pour exprimer le fait que la variable gear peut passer à retracted , représentant ainsi le comportement attendu, ou rester à extended, représentant ainsi le cas de panne ;

- Les événements open et close représentent respectivement l'ouverture et la fermeture de la trappe du train d'atterrissage. Nous remarquons que, de la même manière que dans les événements extend et retract , nous utilisons les substitutions énumérées non-déterministes pour exprimer les cas possibles de panne.

En regardant l'événement de commande du levier PilotCmd, nous remarquons la présence de la clause cmd $\leq$ FCMD dans sa garde. Cette clause implique que cet événement ne peut pas être exécuté plus que FCMD fois sans qu'une séquence ne soit exécutée (exigence E1). L'exigence E2 mentionne que plus le pilote commande le levier, plus la chance d'exécuter l'une des séquences d'extension ou de rétraction augmente. Cette exigence n'est pas prise en compte. Nous avons utilisé des substitutions non-déterministes énumérées pour les actions des événements extend, retract , open et close . Nous expliquons ici l'une de ces substitutions, les autres s'expliquent de la même manière. Dans l'événement close , nous traitons la substitution door $:\in$ {closed , open}. Cette subtitution modélise la panne (door reste à open) mais ne permet pas de préciser la probabilité d'occurence de la panne. Ainsi, l'exigence (E3) n'est qu'en partie prise en compte dans cette machine.

6.2.2 La deuxième machine

Nous appliquons maintenant le processus de probabilisation sur la machine Landing présentée dans la figure 6.4. Nous obtenons la machine Proba−landing−gear donnée dans la figure 6.5.

Cette machine possède les mêmes variables, les mêmes invariants ainsi que les mêmes événements que la machine Landing. Elle utilise les mêmes éléments du contexte contexte0. Dans cette machine, tous les événements sont annotés par des poids.

- L'événement PilotCmd est annoté par le poids FCMD $-$ cmd alors que tous les autres événements sont annotés par le poids FCMD $+$ cmd. À chaque exécution de PilotCmd, la valeur de cmd augmente. Ainsi, le poids de l'événement PilotCmd diminue et les poids des autres événements augmentent. La probabilité d'exécuter l'événement PilotCmd diminue alors progressivement par rapport à la probabilité d'exécuter les autres événements. Nous déduisons ainsi que l'exigence (E2) est bien exprimée dans cette machine contrairement à la machine Landing−gear ;

- Dans les événements extend et retract , la substitution non-déterministe énumérée train $:\in$ {extended, retracted } est remplacée par la substitution train $:=$ extended @ 9/10 $\oplus$ retracted @1/10. Ce qui signifie que la variable train prend la valeur extended avec une probabilité de $\frac{9}{10}$ et la valeur retracted avec une probabilité de $\frac{1}{10}$, ce qui correspond au pourcentage de panne. De la même manière, nous utilisons des substitutions probabilistes énumérées dans tous les autres événements. L'exigence (E3) est bien spécifiée dans l'action des événements correspondants par l'utilisation des substitutions énumérées probabilistes.

Clairement, les expressions des poids des événements de cette machine s'évaluent à des entiers naturels. Ainsi, chacun des événements respecte l'obligation de preuve (evt/WGHT/NAT). À l'exception de l'événement PilotCmd, tous les événements de cette machine possèdent une substitution énumérée probabiliste. À titre d'exemple, l'événement retract possède la substitution

MACHINE
Proba−landing−gear
PROBABILISES
Landing
VARIABLES
handle
door
gear
cmd
INVARIANTS
handle $\in$ HandleState
door $\in$ DoorState
gear $\in$ GearState
cmd $\in \mathbb{N}$
EVENTS
PilotCmd $\widehat{=}$
weight
FCMD $-$ cmd
any
cc
where
cc $\in$ {up,down} $\wedge$ cmd $\leq$ FCMD
then
handle $:=$ cc
cmd $:=$ cmd $+$ 1
end
extend $\widehat{=}$
weight
FCMD $+$ cmd
when
handle $=$ down $\wedge$ door $=$ open $\wedge$ gear $=$ retracted
then
gear $:=$ extended @ 9/10 $\oplus$ retracted @1/10
cmd $:=$ 0
end
event retract
weight
FCMD $+$ cmd
when
handle $=$ up $\wedge$ door $=$ open $\wedge$ gear $=$ extended
then
gear $:=$ retracted @ 9/10 $\oplus$ extended @ 1/10
end
open $\widehat{=}$
weight
FCMD $+$ cmd
when
door $=$ closed $\wedge$ ((handle $=$ down $\wedge$ gear $=$ retracted)
$\vee$ (handle $=$ up $\wedge$ gear $=$ extended))
then
door $:=$ open @ 9/10 $\oplus$ closed @ 1/10
end
.

close $\widehat{=}$
weight
FCMD + cmd
when
door = open $\wedge$ ((handle = down $\wedge$ gear = extended)
$\vee$ (handle = up $\wedge$ gear = retracted))
then
door := closed @ 9/10 $\oplus$ open @ 1/10
end
END

FIGURE 6.5 – Deuxième machine

train := retracted @ 9/10 $\oplus$ extended @ 1/10. Clairement, chacune des valeurs de probabilité dans cette subtitution est entre 0 et 1 et leur somme est égale à 1. Nous concluons ainsi que les deux obligations de preuve (evt/assign/pWD1) et (evt/assign/pWD2) sont bien remplies par cet événement. D'une manière similaire, ces obligations de preuve sont remplies par les événements extend, close et open. L'invariant de cette machine est toujours respecté par tous les événements.

L'utilisation de notre extension probabiliste permet donc au modèle de satisfaire les exigences (E1) et (E2), ce qui n'était pas le cas avec le modèle B événementiel standard.

6.3 Système de freinage

Ce cas d'études représente le système de freinage d'un véhicule. Ce système est en charge de la manoeuvre des freins du véhicule. Dans cette section, nous traitons une version simplifiée de ce système dérivée du projet Deploy [6]. La description du comportement du système est comme suit : le système est géré par le conducteur du véhicule, le conducteur commande le frein par l'intermédiaire d'une pédale. La relation entre la pédale et l'application du frein est la suivante : lorsque le conducteur appuie sur la pédale, le frein doit s'appliquer. Lorsque le conducteur lâche (libère) la pédale, le frein doit se libérer. Le système de freinage doit satisfaire certaines exigences. Dans certains cas, la pédale peut se bloquer : lorsque le conducteur appuie sur la pédale, cette dernière reste alors dans la même position. Cette exigence représente la panne de la pédale. Des mesures externes spécifient que la pédale peut tomber en panne avec un pourcentage de 10%. De la meme manière que la pédale, le frein peut tomber en panne et se bloquer. Dans ce cas, il ne peut pas s'appliquer suite à la commande de la pédale par le conducteur. La panne du frein est liée à son usure. En effet, plus de fois le frein est utilisé, plus la chance de tomber en panne est grande. S'il est appliqué plus qu'un certain nombre de fois, il tombe en panne définitivement.

Nous donnons dans ce qui suit ces exigences d'une manière plus précise.

(E1) Panne de la pédale : la pédale peut rester bloquée en position relachée ou en position appuyée ;

(E2) Risque de la panne de la pédale : la pédale peut tomber en panne avec un pourcentage de 10% ;

(E3) Panne du frein : le frein peut ne pas s'appliquer malgré la commande de la pédale ;

(E4) Usure du frein : plus le frein est appliqué, plus la probabilité de tomber en panne augmente ;

(E5) Le frein ne peut pas s'appliquer plus qu'un certain nombre de fois.

6.3.1 Stratégie du développement du système de freinage en B événementiel

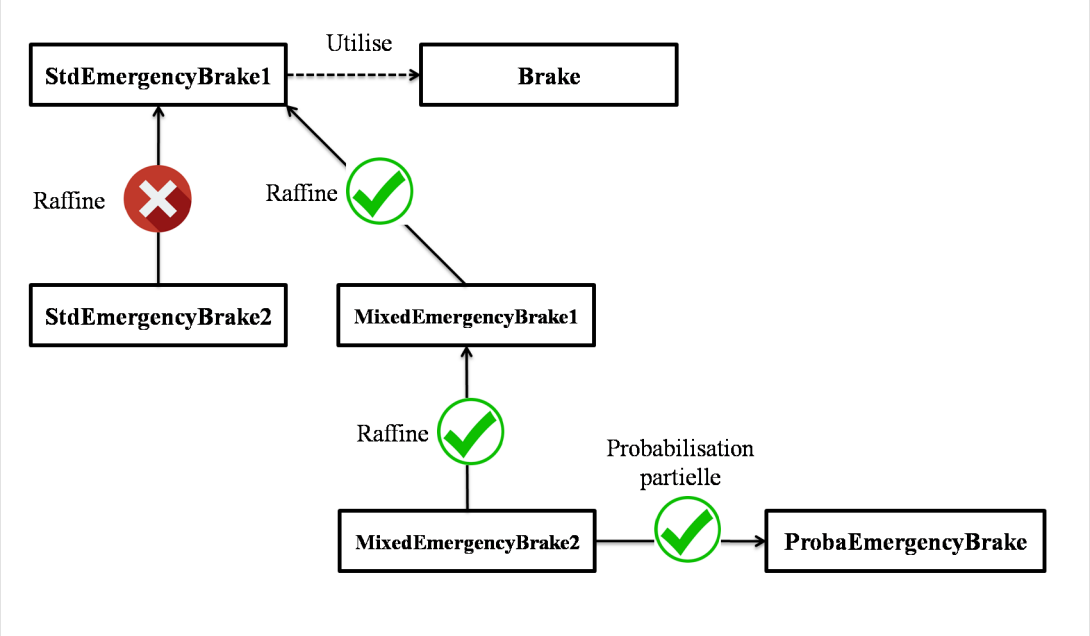

FIGURE 6.6 – Stratégie de développement du système de freinage en B événementiel

Nous présentons dans la figure 6.6 la stratégie de développement adoptée pour la spécification de cette étude de cas. Nous créons un contexte Brake dans lequel nous spécifions les éléments statiques de ce système. Ce contexte est présenté dans la figure 6.7. Nous créons ensuite une machine StdEmergencyBrake1 (figure 6.8). Dans cette machine, nous spécifions le comportement basique de la pédale : l'appui et le relâchement de la pédale par le conducteur. Nous notons que cette machine utilise les éléments du contexte Brake.

Nous proposons un premier raffinement de cette machine, ce raffinement est décrit dans la machine StdEmergencyBrake2 présentée dans la figure 6.9. Nous spécifions l'application du frein ainsi que la panne possible du frein au travers de deux nouveaux événements. Prouver la correction de cette étape de raffinement consiste à prouver la convergence des nouveaux événements introduits. Nous allons montrer que les deux nouveaux événements introduits ne convergent pas car ils ne décroissent pas le variant. Nous proposons alors un nouveau raffinement probabiliste correct de la machine StdEmergencyBrake1, décrit dans la machine MixedEmergencyBrake1 présentée dans la figure 6.10 qui permet de résoudre ce problème. Dans ce nouveau raffinement, nous allons introduire des versions probabilistes des événements précédents et nous allons montrer leur convergence.

Nous raffinons encore une fois la machine MixedEmergencyBrake1 obtenant ainsi la machine MixedEmergencyBrake2 (figure 6.11) dans laquelle nous spécifions le relâchement du frein ainsi que la panne associée. Finalement, nous introduisons les valeurs quantitatives sur le pourcentage de la panne de la pédale dans l'exigence (E2) par probabilisation partielle de la machine MixedEmergencyBrake2. La machine obtenue est ProbEmergencyBrake (figure 6.12).

6.3.2 La première machine

Le système décrit la commande du frein d'un véhicule par l'intermédiaire d'une pédale. La pédale peut avoir deux états : up et down. up représente l'état normal de la pédale lorsque le conduc-

CONTEXT
Brake
CONSTANTS
up
down
applied
released
MAXWEAR
SETS
PedalState
BrakeState
AXIOMS
MAXWEAR $\in$ NAT1
MAXWEAR $>$ 1
partition (PedalState, {up}, {down})
partition (BrakeState, {applied}, {released})
END

FIGURE 6.7 – Contexte initial

teur n'a pas encore appuyé alors que down représente l'état de la pédale après l'appui du conducteur. De même, le frein possède deux états : applied lorsqu'il est appliqué et released lorsqu'il est relâché. Nous créons ainsi deux ensembles : PedalState $=$ {up,down} pour l'état de la pédale et BrakeState $=$ {applied, released} pour l'état du frein. Comme mentionné précédemment, le frein ne peut pas être utilisé plus qu'un certain nombre de fois. Ainsi, nous avons une constante MAXWEAR qui dénote ce nombre maximal d'utilisation du frein.

MACHINE
StdEmergencyBrake1
SEES
Brake
VARIABLES
pedal
INVARIANTS
pedal $\in$ PedalState
EVENTS

Init $\widehat{=}$
then
pedal $:=$ up
end

PushPedal $\widehat{=}$
when
pedal $=$ up
then
pedal $:\in$ {down, up}
end

ReleasePedal $\widehat{=}$
when
pedal $=$ down
then
pedal $:\in$ {up,down}
end

END

FIGURE 6.8 – Première machine

La première machine StdEmergencyBrake1 présentée dans la figure 6.8 spécifie uniquement la manipulation de la pédale par le conducteur. Dans cette machine, l'état du système est décrit par une unique variable : pedal $\in$ {up,down}. Initialement, le conducteur n'a pas encore appuyé sur la pédale, ainsi la variable pedal prend la valeur up. Cette machine contient deux événements : PushPedal et ReleasePedal.

- L'événement PushPedal modélise l'appui du conducteur sur la pédale, c'est-à-dire que la pédale passe de up à down ;

- L'événement ReleasePedal modélise le relâchement de la pédale par le conducteur, c'est-à-dire que la pédale passe de down à up.

Afin de prendre en compte la possibilité d'occurence de pannes lors de l'appui du conducteur sur la pédale exprimée dans l'exigence (E1), nous utilisons une substitution non-déterministe énumérée pedal $:\in$ {up,down} pour exprimer le fait que la pédale peut passer à down représentant ainsi le comportement attendu ou rester bloquée à up représentant ainsi le cas de panne. Une panne est également possible durant le relâchement de la pédale, nous utilisons une substitution non-déterministe énumérée pedal $:\in$ {up,down}, la panne est décrite lorsque la pédale reste bloquée à down. En B événementiel standard, nous ne pouvons pas spécifier les mesures quantitatives concernant la panne de la pédale exprimée par l'exigence (E2).

6.3.3 La deuxième machine

MACHINE
StdEmergencyBrake2
SEES
Brake
VARIABLES
pedal
brake
wear
INVARIANTS
pedal $\in$ PedalState
brake $\in$ BrakeState
wear $\in \mathbb{N}$
wear $\leq$ MAXWEAR
VARIANT
MAXWEAR $-$ wear
EVENTS

Init $\widehat{=}$
then
pedal $:=$ up
brake $:=$ released
wear $:= 0$
end

PushPedal $\widehat{=}$
when
pedal $=$ up
then
pedal $:\in$ {down, up}
end

ReleasePedal $\widehat{=}$
when
pedal $=$ down
then
pedal $:\in$ {up,down}
end

ApplyBrake $\widehat{=}$
when
pedal $=$ down $\wedge$
brake $=$ released $\wedge$
wear $<$ MAXWEAR
then
brake $:=$ applied
wear $:=$ wear$+1$
end

ApplyBrakeFailure $\widehat{=}$
when
pedal $=$ down $\wedge$
brake $=$ released $\wedge$
wear $<$ MAXWEAR
then
brake $:=$ released
end

END

FIGURE 6.9 – Deuxième machine

La deuxième machine StdEmergencyBrake2 décrite en figure 6.9 raffine StdEmergencyBrake1. Cette machine utilise le même contexte Brake que StdEmergencyBrake1. Nous avons ajouté deux nouvelles variables dans cette machine : brake et wear. brake décrit l'état du frein (brake $\in$ {applied, released}) et wear (wear $\in$ NAT) représente le nombre de fois où le frein a été appliqué. Initialement, le conducteur n'a pas encore appuyé sur la pédale et le frein n'a jamais été appliqué. Ainsi, la variable brake est initialisée à released et wear est initialisée à 0.

Nous ajoutons deux nouveaux événements : ApplyBrake et ApplyBrakeFailure.

- L'événement ApplyBrake modélise l'application du frein : la variable brake prend la valeur applied et la variable wear est incrémentée de 1. La garde de cet événement wear $<$ MAXWEAR impose une contrainte sur l'application du frein : le frein ne peut pas être appliqué plus qu'un certain nombre de fois, ici MAXWEAR. Ainsi, l'exigence (E5) est bien respectée dans cette machine.

- L'événement ApplyBrakeFailure modélise la panne possible durant l'application du frein : le conducteur commande l'application du frein et ce dernier ne s'applique pas (la variable brake reste à la valeur released). Ainsi, l'exigence (E3) sur la panne du frein est bien modélisée.

Afin de montrer la correction de cette étape de raffinement, nous devons montrer que les deux événements ApplyBrake et ApplyBrakeFailure convergent. Nous proposons un variant MAXWEAR $-$ wear de type entier. Il est bien minoré par 0, wear étant inférieur ou égale à MAXWEAR ((evt/-var/NAT)). Nous devons également montrer que les événements ApplyBrake et ApplyBrakeFailure décroîssent le variant.

- À chaque exécution de l'événement ApplyBrake, la valeur de wear augmente et ainsi, la valeur du variant diminue.

- L'action de l'événement ApplyBrakeFailure ne modifie pas la valeur de la variable wear, et ainsi, cet événement ne diminue pas la valeur du variant. L'obligation de preuve (evt/VAR) n'est pas respectée par cet événement.

Cette étape de raffinement n'est pas correcte. Nous montrons dans ce qui suit l'utilité de notre proposition de B événementiel probabiliste pour faciliter la preuve de convergence de nouveaux événements probabilistes introduits par raffinement.

6.3.4 La troisième machine : version probabiliste

Dans la machine précédente, nous avons introduit deux nouveaux événements ApplyBrake et ApplyBrakeFailure. Cependant, nous avons montré que l'ensemble de ces deux événements ne converge pas puisque l'événement ApplyBrakeFailure ne décroît pas la valeur du variant.

Nous présentons dans la figure 6.10 une version mixte de la machine StdEmergencyBrake2. Nous introduisons une version probabiliste des événements ApplyBrake et ApplyBrakeFailure.

L'événement ApplyBrake est maintenant annoté par le poids MAXWEAR $-$ wear alors que l'événement
ApplyBrakeFailure est annoté par le poids wear. Nous constatons qu'à chaque exécution de l'événement ApplyBrake, la valeur de wear augmente. Ainsi, le poids de l'événement ApplyBrake diminue et le poids de ApplyBrakeFailure augmente. La probabilité d'exécuter l'événement ApplyBrake diminue au fur et à mesure de son exécution alors que la probabilité d'exécuter l'événement ApplyBrakeFailure augmente. Ce qui correspond à l'exigence (E4). Concernant la décroissance du variant, puisque les événements ApplyBrake et ApplyBrakeFailure sont probabilistes et sont toujours activables dans la même valuation des variables, il suffit que l'un de ces événements diminue la valeur du variant (machine/pVAR), ce qui est le cas pour l'événement ApplyBrake. Ainsi, nous pouvons prouver la convergence presque certaine de ces deux événements lorsqu'ils sont probabilistes contrairement au cas où ils sont non-déterministes.

Nous remarquons l'intérêt de notre extension probabiliste au B événementiel dans la preuve de convergence de nouveaux événements introduits par raffinement.

MACHINE
MixedEmergencyBrake1
SEES
Brake
VARIABLES
pedal
brake
wear
INVARIANTS
pedal $\in$ PedalState
brake $\in$ BrakeState
wear $\in \mathbb{N}$
wear $\leq$ MAXWEAR
VARIANT
MAXWEAR $-$ wear
EVENTS
Init $\widehat{=}$
pedal := up
brake := released
wear := 0
end

PushPedal $\widehat{=}$
when
pedal $=$ up
then
pedal $:\in$ {down, up}
end

ReleasePedal $\widehat{=}$
when
pedal $=$ down
then
pedal $:\in$ {up,down}
end

ApplyBrake $\widehat{=}$
weight
MAXWEAR $-$ wear
when
pedal $=$ down $\wedge$ brake $=$ released $\wedge$ wear $<$ MAXWEAR
then
brake := applied
wear := wear+1
end

ApplyBrakeFailure $\widehat{=}$
weight
wear
when
pedal $=$ down $\wedge$ brake $=$ released $\wedge$ wear $<$ MAXWEAR
then
brake := released
end

END

FIGURE 6.10 – Troisième machine

6.3.5 La quatrième machine

La quatrième machine MixedEmergencyBrake2 présentée dans la figure 6.11 raffine la troisième machine MixedEmergencyBrake1 présentée dans la figure 6.10. Cette machine contient les mêmes variables, les mêmes invariants ainsi que le même variant que la machine MixedEmergencyBrake1. Dans cette machine, nous introduisons deux nouveaux événements probabilistes modélisant le relâchement du frein : ReleaseBrake et ReleaseBrakeFailure.

L'événement ReleaseBrake modélise le relâchement du frein, c'est-à-dire que la variable brake prend la valeur released et la variable wear est incrémentée de 1. L'événement ReleaseBrakeFailure modélise la panne qui peut avoir lieu durant le relâchement du frein, c'est-à-dire que la variable brake reste à applied.

Ces deux événements sont deux nouveaux événements probabilistes introduits par raffinement. Comme précédemment, nous devons prouver leur convergence. Ces deux événements sont activables dans la même valuation des variables et l'événement ReleaseBrake diminue la valeur du variant (puisque son action augmente la valeur du wear). Ainsi, nous déduisons que l'obligation de preuve ((machine/pVAR)) est toujours respesctée et ces deux nouveaux événements convergent.

6.3.6 La cinquième machine

Nous présentons dans la figure 6.12 la machine ProbEmergencyBrake. Cette machine possède les mêmes variables, les mêmes invariants ainsi que les mêmes événements que la machine

MACHINE
MixedEmergencyBrake2
SEES
Brake
VARIABLES
pedal
brake
wear
INVARIANTS
pedal $\in$ {up,down}
brake $\in$ {applied , released }
wear $\in \mathbb{N}$
wear $\leq$ MAXWEAR
VARIANT
MAXWEAR $-$ wear
EVENTS

Init $\widehat{=}$
then
pedal := up
brake := released
wear := 0
end

PushPedal $\widehat{=}$
when
pedal $=$ up
then
pedal :$\in$ {down, up}
end

ReleasePedal $\widehat{=}$
when
pedal $=$ down
then
pedal :$\in$ {up,down}
end

ApplyBrake $\widehat{=}$
weight
MAXWEAR $-$ wear
when
pedal $=$ down $\wedge$
brake $=$ released $\wedge$ wear $<$ MAXWEAR
then
brake := applied
wear := wear+1
end

ApplyBrakeFailure $\widehat{=}$
weight
wear
when
pedal $=$ down $\wedge$ brake $=$ released $\wedge$ wear $<$ MAXWEAR
then
brake := released
end

ReleaseBrake $\widehat{=}$
weight
MAXWEAR $-$ wear
when
pedal $=$ up $\wedge$ brake $=$ applied $\wedge$ wear $<$ MAXWEAR
then
brake := released
wear := wear+1
end

ReleaseBrakeFailure $\widehat{=}$
weight
wear
when
pedal $=$ up $\wedge$ brake $=$ applied $\wedge$ wear $<$ MAXWEAR
then
brake := applied
end

END

FIGURE 6.11 – Quatrième machine

MixedEmergencyBrake2. Elle utilise le même contexte Brake. Dans cette machine, nous appliquons la probabilisation partielle sur les événements PushPedal et ReleasePedal. Les substitutions non-déterministes de ces deux événements sont remplacées par des substitutions probabilistes énumérées. Par exemple, la substitution pedal :$\in$ {down,up} de l'événement PushPedal est remplacée par la substitution pedal := down @9/10 $\oplus$ up @1/10. La variable pedal prend la valeur down avec une probabilité de $\frac{9}{10}$ et la valeur up avec une probabilité de $\frac{1}{10}$, qui est la probabilité de la panne. De la même manière, nous utilisons le même type de substitutions dans l'événement ReleasePedal. Ainsi, l'exigence (E2) est bien prise en compte.

À ce niveau de raffinement, toutes les exigences sont prises en compte.

Nous considérons la machine ProbaEmergencyBrake. Clairement, les expressions des poids des événements de cette machine s'évaluent à des entiers naturels. Ainsi, chacun des événements

MACHINE
ProbaEmergencyBrake
SEES
Brake
VARIABLES
pedal
brake
wear

INVARIANTS
pedal $\in$ {up,down}
brake $\in$ {applied, released}
wear $\in \mathbb{N}$
wear $\leq$ MAXWEAR

EVENTS
Init $\widehat{=}$
then
pedal := up
brake := released
wear := 0
end

PushPedal $\widehat{=}$
weight
MAXWEAR
when
pedal = up
then
pedal := down @9/10 $\oplus$ up @1/10
end

ReleasePedal $\widehat{=}$
weight
MAXWEAR
when
pedal = down
then
pedal := up @9/10 $\oplus$ up @1/10
end

ApplyBrake $\widehat{=}$
weight
MAXWEAR $-$ wear
when
pedal = down $\wedge$
brake = released $\wedge$ wear $<$ MAXWEAR
then
brake := applied
wear := wear+1
end

ReleaseBrake $\widehat{=}$
weight
MAXWEAR $-$ wear
when
pedal = up $\wedge$ brake = applied $\wedge$ wear $<$ MAXWEAR
then
brake := released
end

ApplyBrakeFailure $\widehat{=}$
weight
wear
when
pedal = down $\wedge$ brake = released $\wedge$ wear $<$ MAXWEAR
then
brake := released
end

ReleaseBrakeFailure $\widehat{=}$
weight
wear
when
pedal = up $\wedge$ brake = applied $\wedge$ wear $<$ MAXWEAR
then
brake := released
end

END

FIGURE 6.12 – Machine probabiliste

respecte l'obligation de preuve (evt/WGHT/NAT). L'événement PushPedal possède une substitution probabiliste énumérée pedal := down @ 9/10 $\oplus$ up @ 1/10. Les deux obligations de preuve (evt/assign/pWD1) et (evt/assign/pWD2) sont donc bien remplies par cet événement. L'invariant de cette machine est toujours respecté par tous les événements de cette machine, ce qui conclue la preuve de cohérence.

7

Extension de la plateforme Rodin

7.1 Introduction

La méthode B événementiel est équippée de la plateforme Rodin [14, 19, 36, 10], une plateforme qui permet la spécification ainsi que la preuve de correction des machines B événementiel. Dans les chapitres précédents de ce manuscrit, nous avons présenté la méthode B événementiel ainsi que notre extension probabiliste à cette méthode afin de prendre en considération la modélisation des comportements probabilistes. Afin d'outiller notre extension probabiliste au B événementiel, nous avons commencé le développement d'une extension à Rodin afin de prendre en compte nos propositions. Dans la section 7.2, nous présentons la plateforme Rodin, son architecture ainsi que son principe de fonctionnement. Dans la section 7.3, nous présentons les éléments de notre extension.

7.2 La plateforme Rodin

Rodin [14, 19, 36, 10] est un outil libre qui permet la spécification de modèles B événementiel ainsi que la preuve de leur correction. Cet outil a été développé dans le cadre des projets européens *Rodin* [11] puis *Deploy* [6]. C'est un outil à base de plugins qui s'exécutent dans l'environnement de développement Eclipse [7]. La composition à base de plugins fait de Rodin un outil de modélisation et de preuve facilement extensible et configurable. De nombreux plugins peuvent être intégrés à Rodin. Nous citons à titre d'exemple les plugins ProB, UML-B et B2Latex. ProB [73, 9] est un plugin qui permet la vérification de la cohérence des machines B événementiel par *model checking*. UML-B [88, 12] est un plugin permettant la transformation des modèles UML en modèles B événementiel. B2Latex [2] est un plugin permettant la génération de fichiers Latex correspondant à une spécification B événementiel.

7.2.1 Architecture de Rodin

La figure 7.1 présente l'architecture générale de Rodin. Comme présenté dans cette figure, Rodin est une extension de la plateforme Eclipse. L'architecture de Rodin est divisée en quatre grandes parties, le noyau Rodin (*Rodin Core*), les librairies B événementiel (*Event-B Library packages*), le noyau B événementiel (*Event-B Core*) et l'interface graphique (*Event-B UI*). Nous décrivons brièvement dans ce qui suit chacune de ces parties :

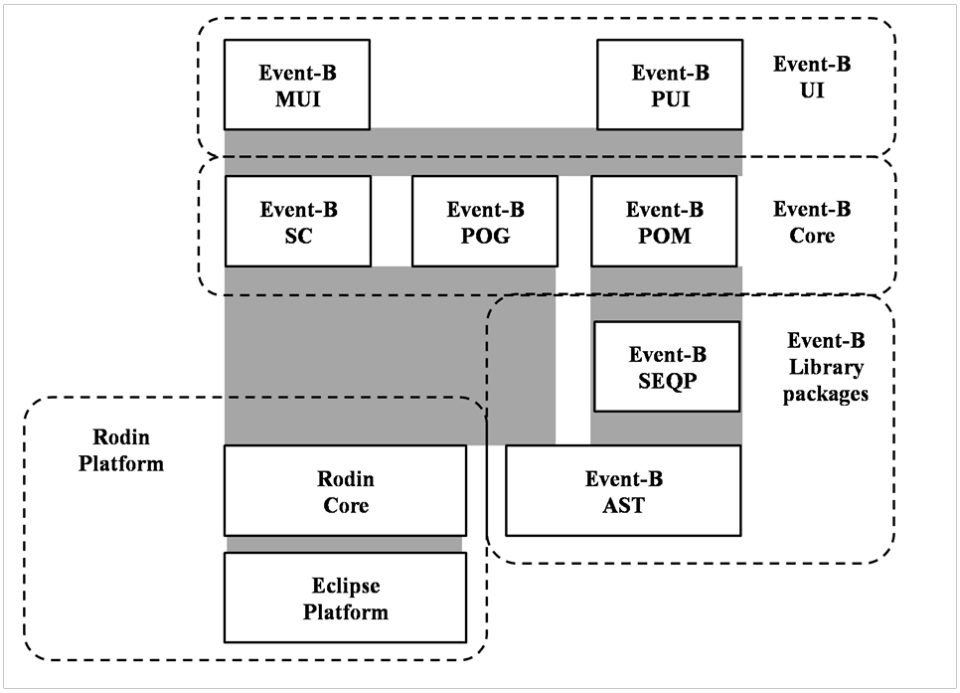

FIGURE 7.1 – Architecture de Rodin

Le noyau Rodin (*Rodin Core*) Le noyau Rodin est constitué de deux composants de base : le dépôt Rodin (*RODIN Repository*) et le constructeur Rodin (*RODIN builder*). Ces deux composants sont intégrés à Eclipse et sont indépendants du B événementiel. Le dépôt Rodin assure la persistance des éléments de base de Rodin (syntaxe, opérateurs, types, modèles, projets..). L'utilisation d'un dépôt Rodin à la place d'une syntaxe fixe pour les notations de modélisation facilite l'extension du B événementiel : il n'est plus nécessaire de changer la syntaxe. Le constructeur Rodin s'occupe de l'ordonnancement des tâches en fonction des modifications qui sont faites sur les fichiers contenus dans le dépôt Rodin.

Les librairies B événementiel (*Event-B Library Packages*) Dans Rodin, les modèles B événementiel n'ont pas de syntaxe précise, ils sont enregistrés dans des dépôts. Cependant, les notations mathématiques utilisées (les invariants ou les gardes) ont une syntaxe précise et elles sont spécifiées par une grammaire précise enregistrée dans ces librairies. De même, tous les éléments du moteur de preuve, à savoir, les types de données, les règles d'inférences, les tactiques de support de preuve sont stockés dans une librairie spécifique contenue dans cette partie.

Le noyau B événementiel (*Event-B Core*) Le noyau du B événementiel est formé de trois composants : l'analyseur de syntaxe (*Static checker (SC)*), le générateur d'obligations de preuve (*proof obligation generator (POG)*), et le gestionnaire des obligations de preuve (*Proof obligation manager (POM)*). Nous les détaillons dans ce quit suit :

- L'analyseur de syntaxe analyse les contextes et les machines d'un modèle B événementiel et dégage les éventuelles erreurs de syntaxe et de typage statique. Il rejette les éléments de données qui ne satisfont pas la grammaire du B événementiel et produit des messages d'erreurs. L'analyseur de syntaxe peut être étendu par l'ajout de nouveaux éléments ou par la suppression d'autres éléments ;

- Le générateur d'obligations de preuve génère les obligations de preuve d'un modèle validé par l'analyseur syntaxique ;

- Le gestionnaire des obligations de preuve assure la démonstration des obligations de preuve d'un modèle B événementiel. Pour les obligations de preuve nécessitant l'intervention de l'utilisateur, il offre à ce dernier une interface interactive de preuve. Le gestionnaire d'obligations de preuve offre principalement trois fonctionnalités :

  – faire le lien entre les obligations de preuve standard et les obligations de preuves modifiées par l'utilisateur,
  – démontrer les obligations de preuve automatiquement lorsque c'est possible,
  – fournir une interface pour la preuve interactive.

L'ordonnancement des tâches entre ces trois composants est assuré par le constructeur Rodin. La connection entre ces trois composants est montrée dans la figure 7.2.

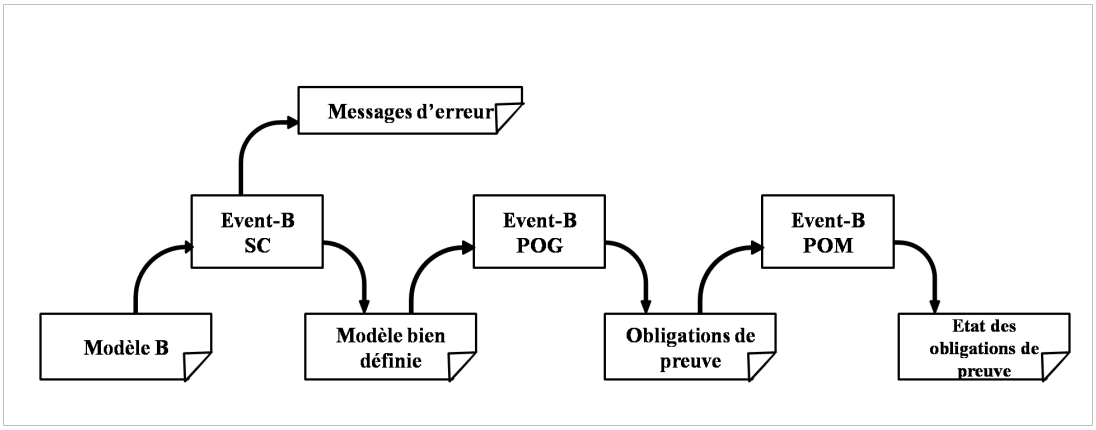

FIGURE 7.2 – Interaction entre les composants du noyau B événementiel

L'interface graphique (*Graphical User Interface*) L'interface graphique de Rodin est constituée de deux parties : une interface pour la modélisation et une interface pour la preuve. La figure 7.3 montre l'interface de modélisation. La figure 7.4 donne un aperçu sur l'interface de preuve dans Rodin. La figure 7.5 montre l'interaction entre les composants du noyau B événementiel et l'interface graphique de Rodin. Nous notons que l'interface graphique de preuve n'accède pas directement aux obligations de preuve et aux preuves mais qu'elle utilise les services offerts par le gestionnaire d'obligations de preuve. Nous constatons que les interfaces de modélisation et de preuve sont connectées par les composants du noyau B événementiel de Rodin.

7.2.2 Processus de développement d'un modèle B événementiel dans Rodin

Nous présentons brièvement le processus de développement d'un modèle B événementiel dans Rodin. Dans un premier temps, l'utilisateur écrit sa spécification. Une fois terminé, Rodin lance l'analyseur de syntaxe pour vérifier les contextes et les machines du modèle. S'il détecte des erreurs, il les affiche à l'utilisateur. Dans le cas contraire, c'est le générateur d'obligations de preuve qui prend le contrôle en générant automatiquement les obligations de preuve spécifiques au modèle. Ces obligations de preuve constituent, une fois démontrées, une preuve de correction du modèle. La plupart des obligations de preuve sont déchargées automatiquement par les prouveurs de Rodin. Cependant, dans certains cas, la démonstration de certaines obligations de preuve nécessite l'intervention de l'utilisateur. Cela peut être dû à une obligation de preuve dont la forme n'est pas simple et qui nécessite donc une étude plus détaillée et guidée par l'utilisateur ou au cas

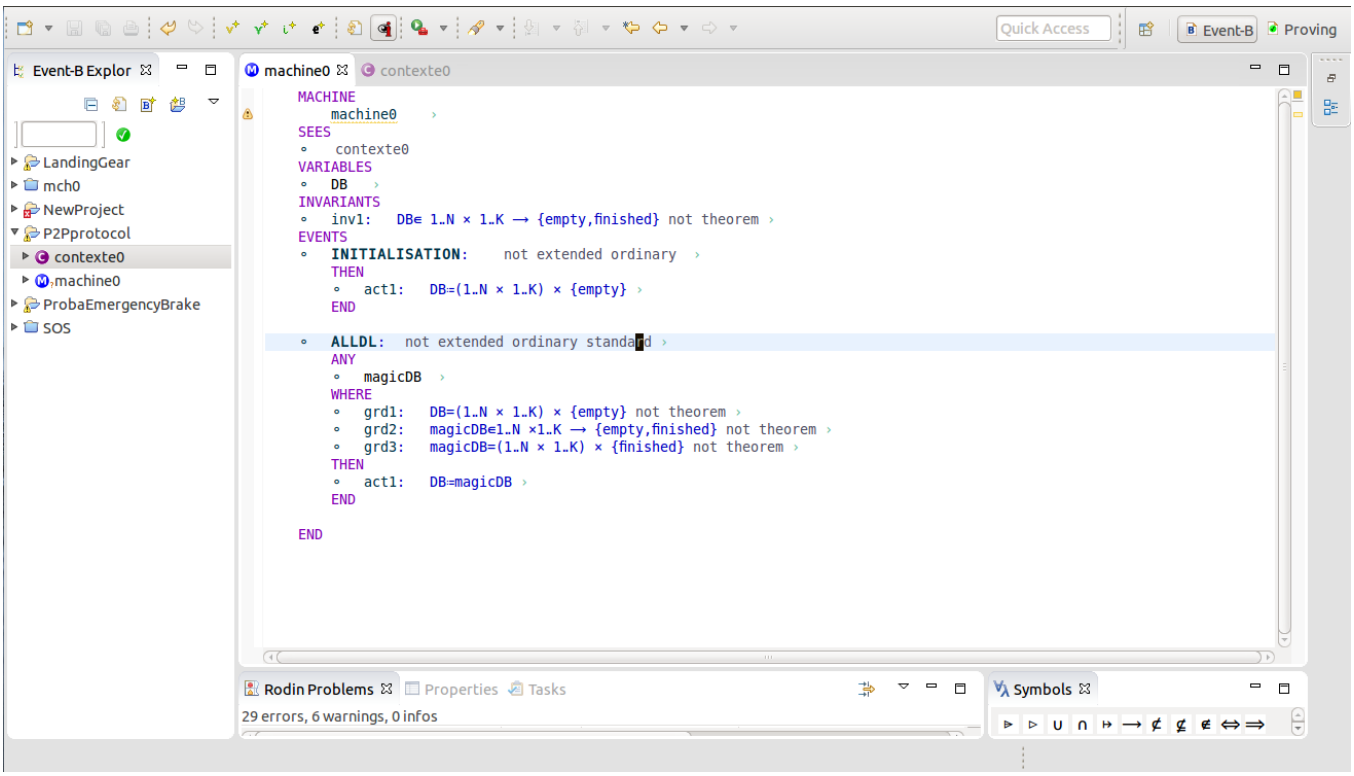

FIGURE 7.3 – Interface de modélisation dans l'outil Rodin

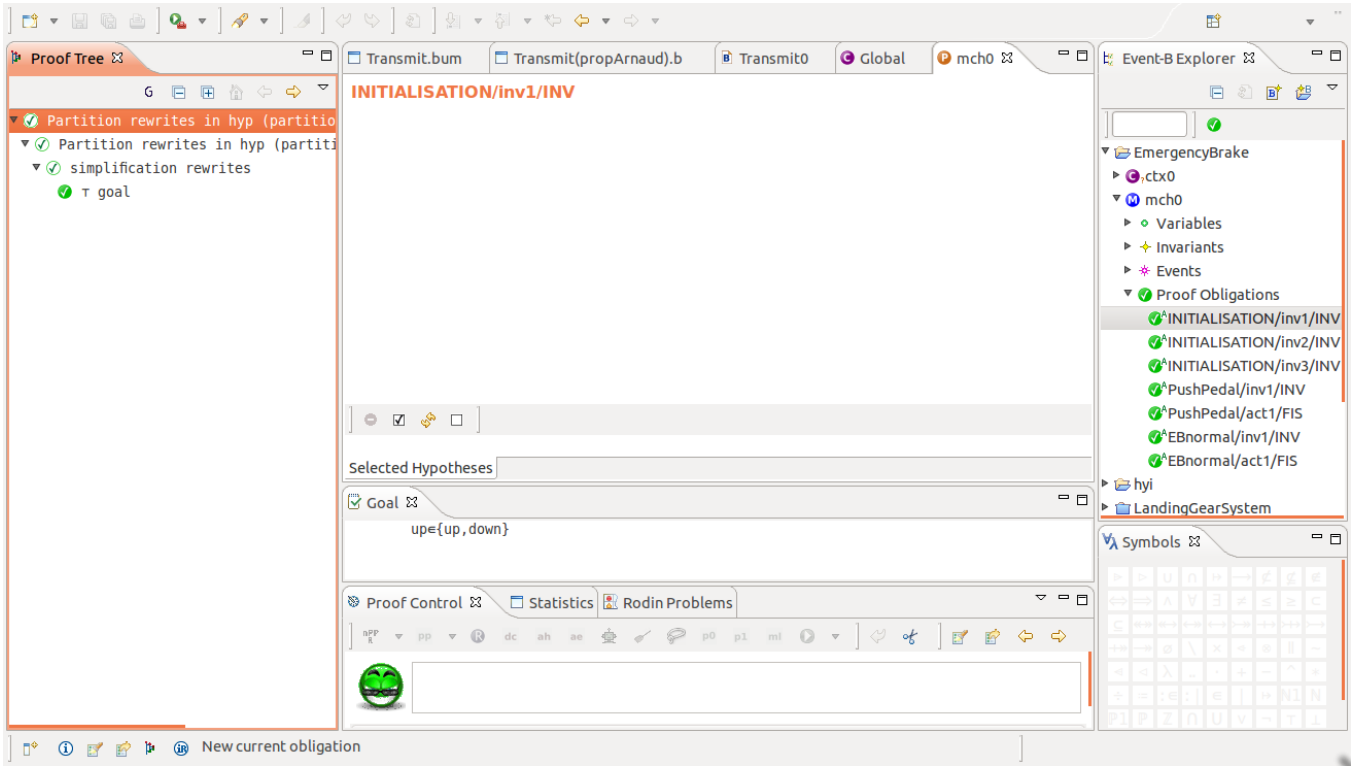

FIGURE 7.4 – Interface de preuve dans l'outil Rodin

où la preuve ne peut être établie parce que le modèle comporte des erreurs. À la fin du processus, un modèle B événementiel est correct si et seulement si toutes les obligations de preuve sont démontrées.

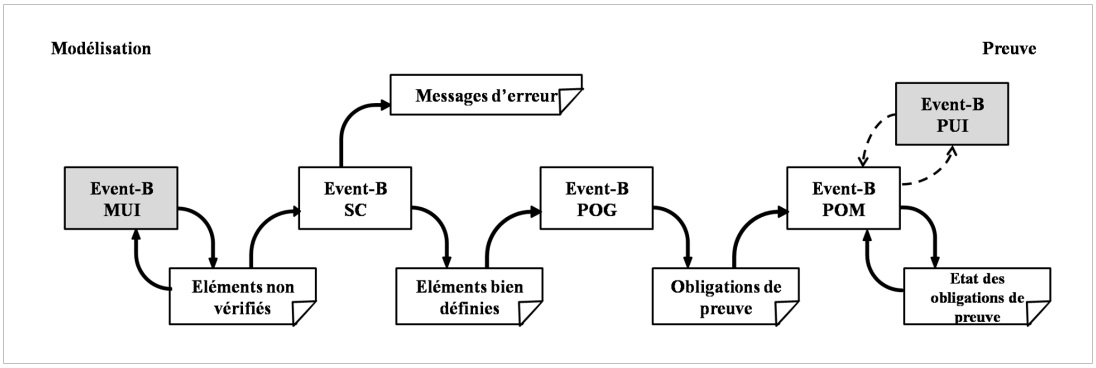

FIGURE 7.5 – Interaction entre les interfaces de Rodin et le noyau B événementiel

7.3 Intégration de notre extension probabiliste dans Rodin

Dans cette thèse, nous avons proposé une extension probabiliste au B événementiel permettant la prise en compte de la modélisation ainsi que de la preuve des comportements probabilistes en B événementiel. Afin de pouvoir outiller notre extension, nous avons commencé le développement d'une extension à Rodin permettant l'intégration des éléments ajoutés à B événementiel. Nous notons qu'à l'heure actuelle, cette extension n'est pas finalisée et ne prend pas encore en compte tous les éléments de notre proposition de B événementiel probabiliste.

7.3.1 Description de l'extension

Rodin est une plateforme à base de plugins, elle utilise le mécanisme d'extensibilité d'Eclipse. Ainsi, nous pouvons ajouter de nouvelles fonctionnalités dans Rodin par ajout de nouveaux plugins. Dans notre proposition de B événementiel probabiliste, nous avons introduit de nouveaux éléments de syntaxe. De même, nous avons ajouté de nouvelles obligations de preuve et nous avons adapté les obligations de preuve standard pour qu'elles puissent être utilisées dans notre proposition. Nous avons également proposé deux nouveaux processus de développement en B événementiel, *la probabilisation* et *la probabilisation partielle*. Dans notre extension, nous avons étendu les librairies B événementiel, le noyau B événementiel ainsi que l'interface graphique de modélisation. Nous présentons dans ce qui suit quelques éléments de notre extension.

Annotation d'un événement par un poids dans Rodin Nous présentons ici notre extension à la plateforme Rodin permettant d'annoter un événement par un poids.

Nous avons étendu l'éditeur de Rodin afin d'offrir à l'utilisateur la possibilité d'ajouter un poids à un événement. Pour cela, nous avons étendu le noyau Rodin, l'analyseur de syntaxe ainsi que l'interface graphique de modélisation. La figure 7.6 montre cette extension à l'éditeur de Rodin. Voulant ajouter un élément de modélisation à l'événement ALLDL, l'éditeur propose à l'utilisateur en plus des éléments standards, la possibilité d'ajouter un poids.

La figure 7.7 montre l'annotation d'un événement par un poids. Nous remarquons la nouvelle clause **WEIGHT** et l'annotation de l'événement ALLDL par le poids $N \times k$. Nous notons que initialement, à l'ajout de la clause **WEIGHT**, le champ pour le poids est vide, c'est à l'utilisateur de saisir la valeur du poids. Pour cette extension, nous avons étendu le noyau Rodin, les librairies B événementiel, l'analyseur de syntaxe ainsi que l'interface graphique.

Ajout d'une substitution probabiliste La palateforme Rodin offre à l'utilisateur la possibilité d'ajouter des substitutions non-déterministes et des substitutions déterministes. Afin de prendre

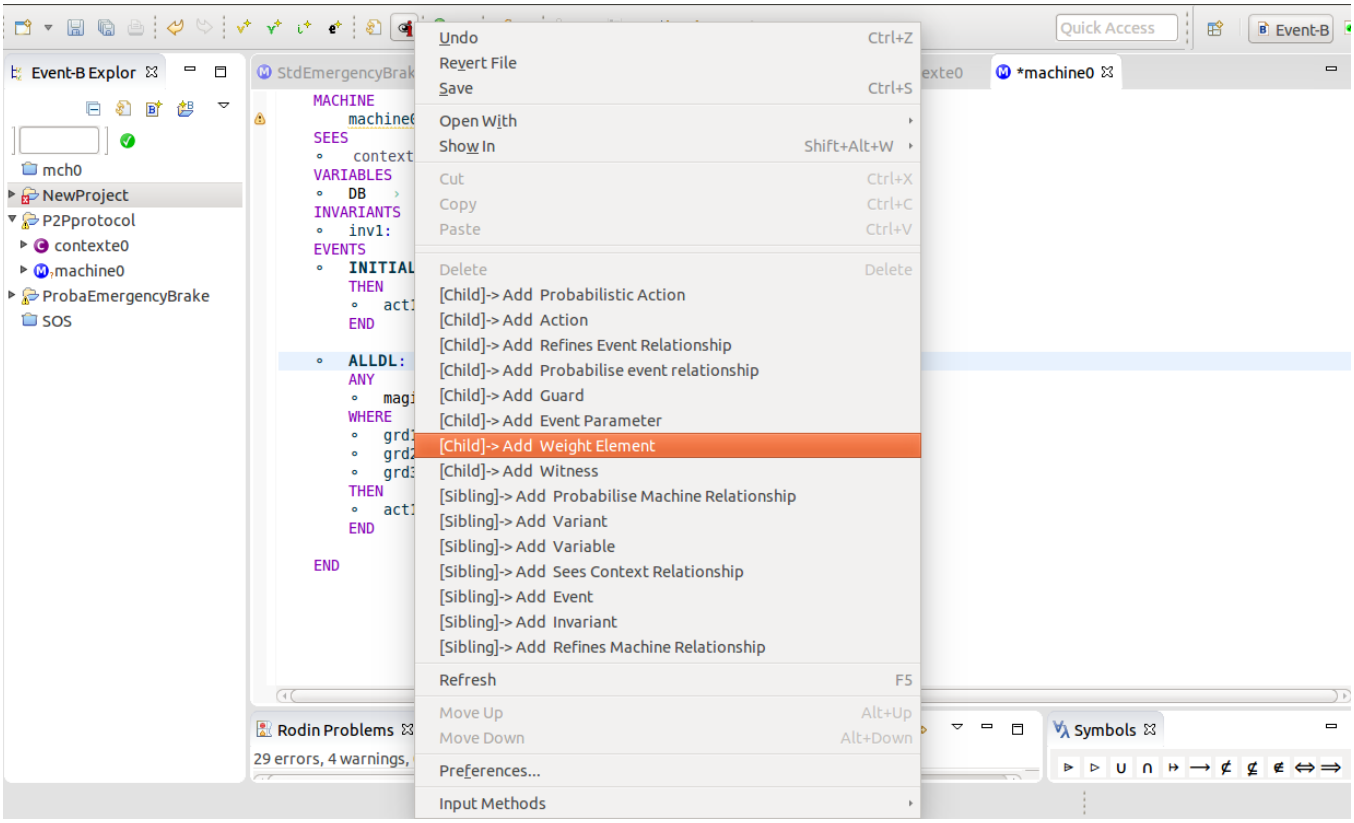

FIGURE 7.6 – Ajout d'un poids à un événement

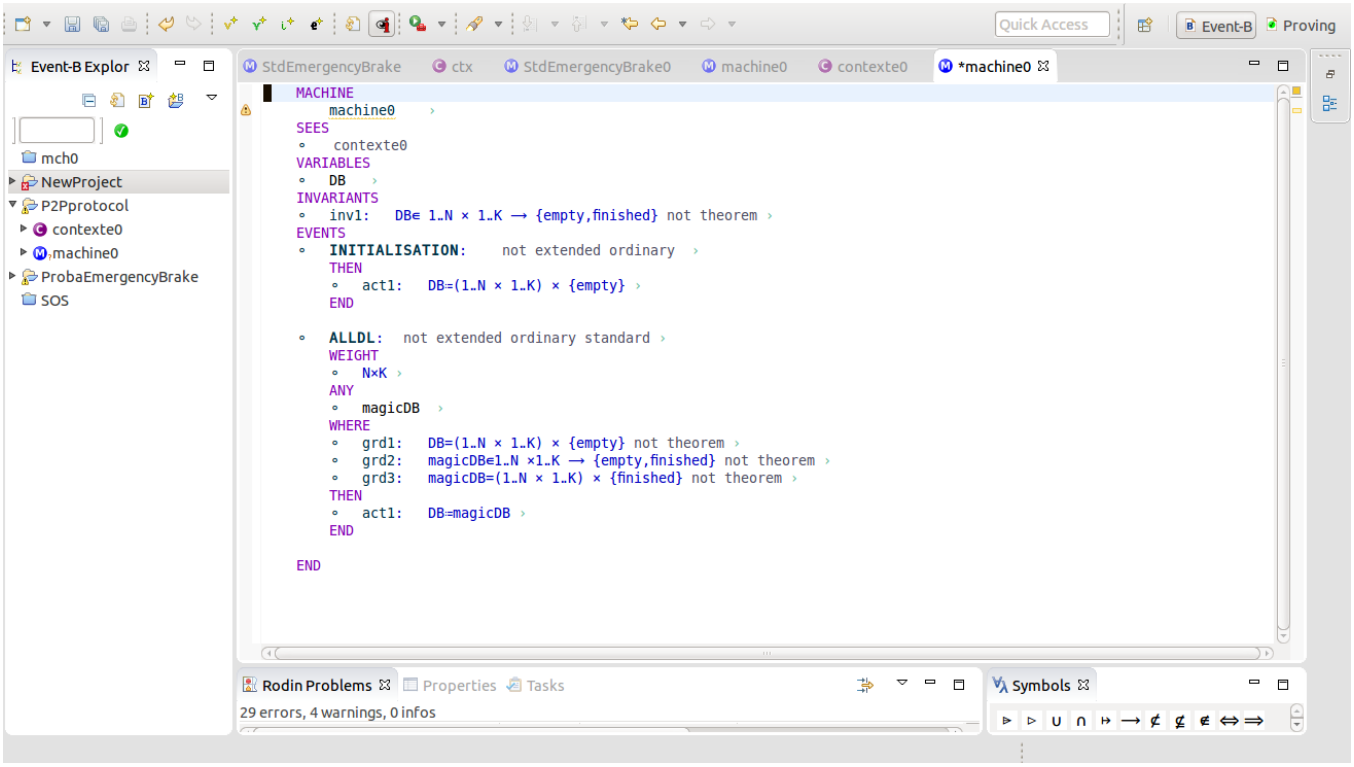

FIGURE 7.7 – Annotation d'un événement par un poids

en compte les substitutions énumérées probabilistes de notre extension, nous avons étendu l'éditeur de Rodin pour offrir à l'utilisateur la possibilité d'ajouter à un événement une substitution probabiliste énumérée. La figure 7.8 montre cette extension à l'éditeur de Rodin.

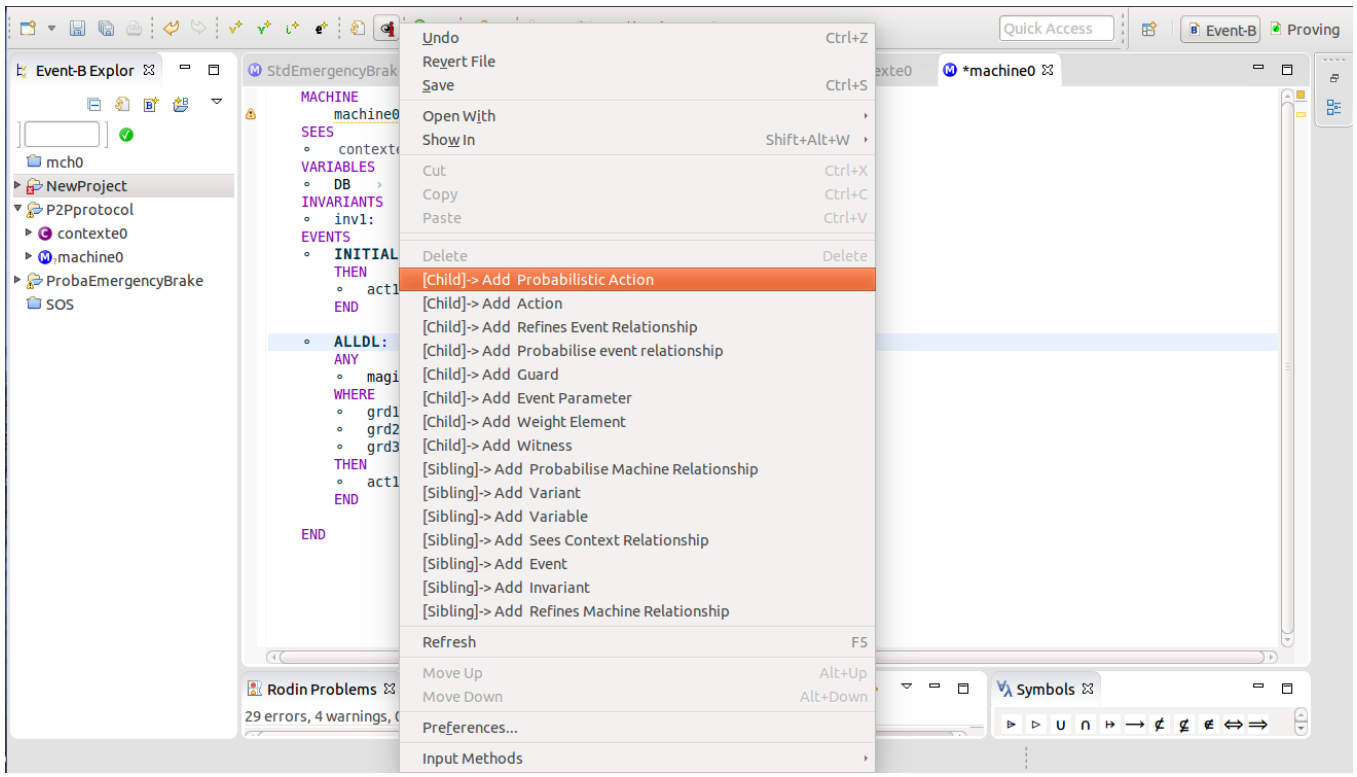

FIGURE 7.8 – Ajout d'une substitution probabiliste

Comme dans le cas de possibilité d'ajout d'un poids à un événement, pour réaliser cette extension, nous avons étendu le noyau Rodin, les librairies B événementiel, l'analyseur de syntaxe ainsi que l'interface graphique.

Intégration du processus de probabilisation Notre extension à Rodin prend aussi en charge le processus de probabilisation. Nous reprenons ici le cas de la machine LandingGear présentée dans la section 6. La figure 7.9 montre cette machine décrite en B événementiel standard dans Rodin.

Nous avons ajouté dans l'éditeur de Rodin une nouvelle opération qui permet de probabiliser une machine B événementiel non-déterministe. La figure 7.10 montre cette extension permettant d'appliquer la probabilisation. Une nouvelle opération dénotée par PROBABILISE est offerte à l'utilisateur. Lorsque l'utilisateur sélectionne PROBABILISE, une interface graphique s'affiche permettant de préciser (mentionner) le nom de la machine résultante de la probabilisation, l'utilisateur saisit le nom de la machine résultante et Rodin génére cette machine. La figure 7.11 montre l'interface permettant la saisie du nom de la machine.

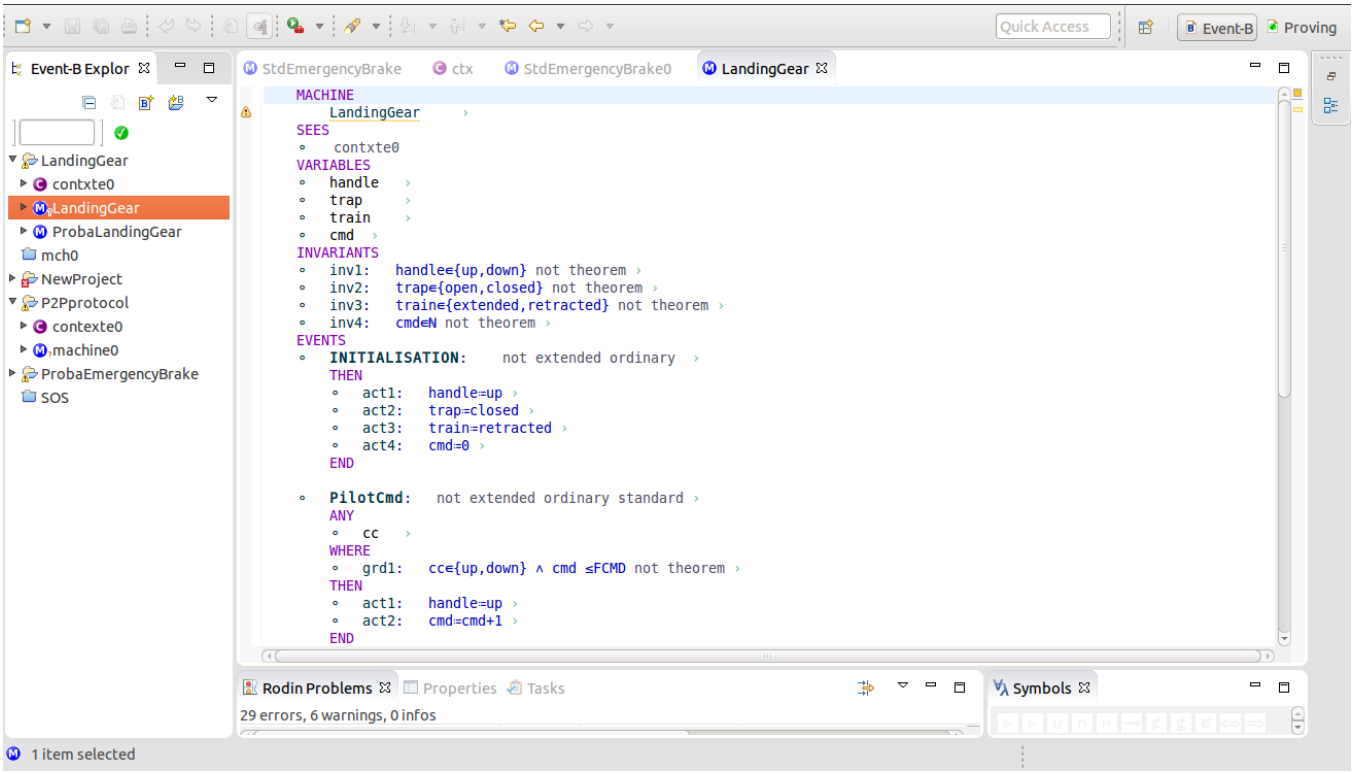

FIGURE 7.9 – Machine non-déterministe du LandingGear

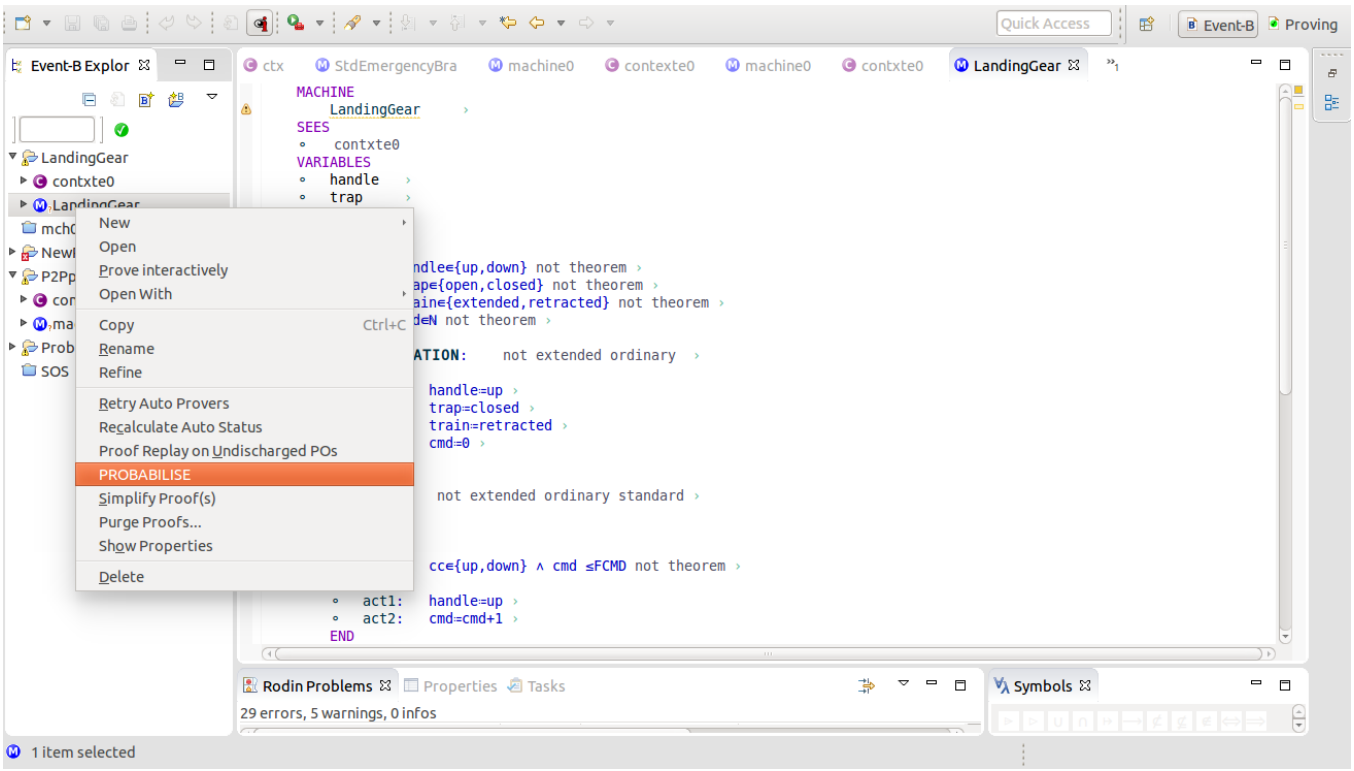

FIGURE 7.10 – Application du processus de probabilisation

La figure 7.12 montre la machine résultante de cette opération.

Dans cette machine, nous remarquons l'apparition de la clause PROBABILISE qui indique la machine probabilisé. De même, nous remarquons l'apparition de cette clause dans chaque événement, elle permet de préciser le nom de l'événement probabilisé par l'événement correspondant. Nous remarquons que le type de tous les événements est probabiliste indiqué par le mot clé probabilistic .

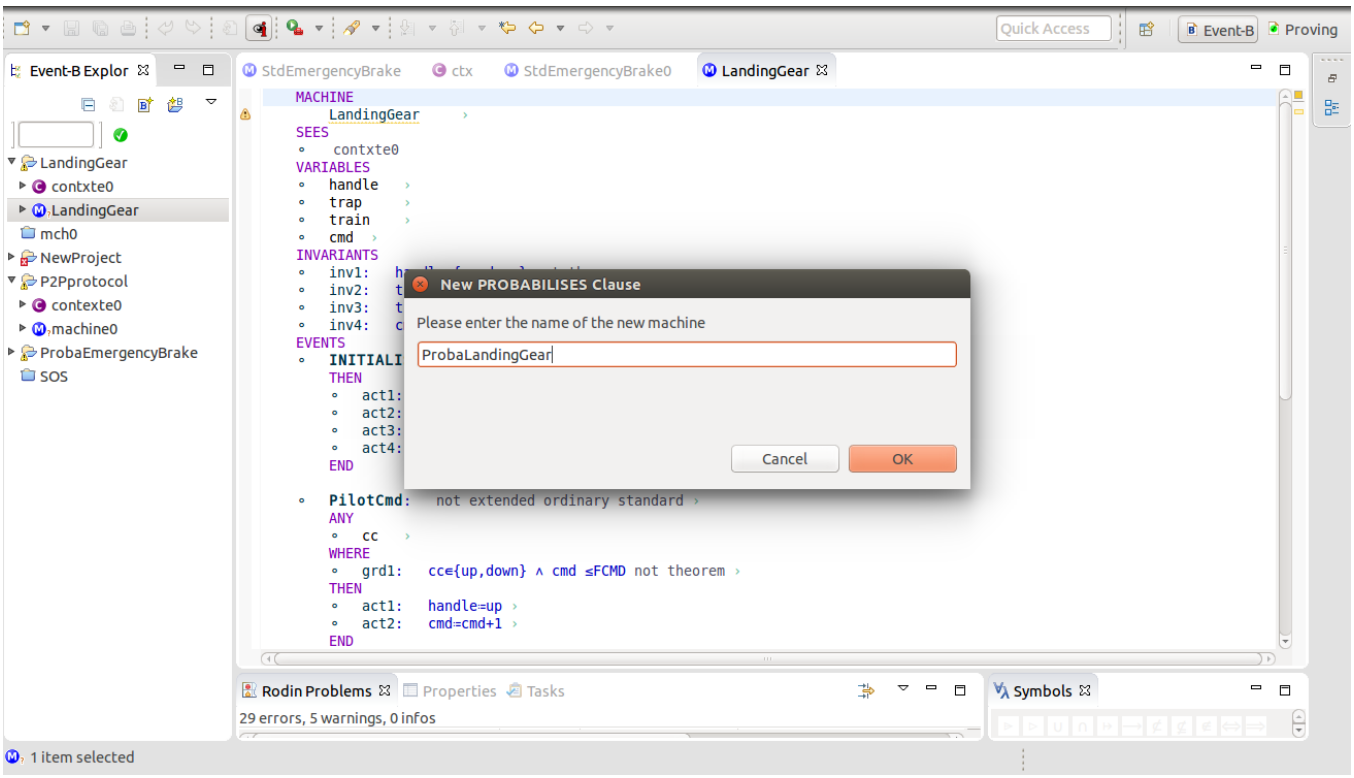

FIGURE 7.11 – Précision du nom de la machine

Nous remarquons aussi l'annotation de tous les événements par des poids ainsi que l'apparition des substitutions probabilistes. À la génération de la machine, les champs pour les valeurs des poids des événements ainsi que les valeurs de probabilité dans les substitutions probabilistes ne sont pas précisées, c'est à l'utilisateur de saisir les valeurs correspondantes selon son besoin.

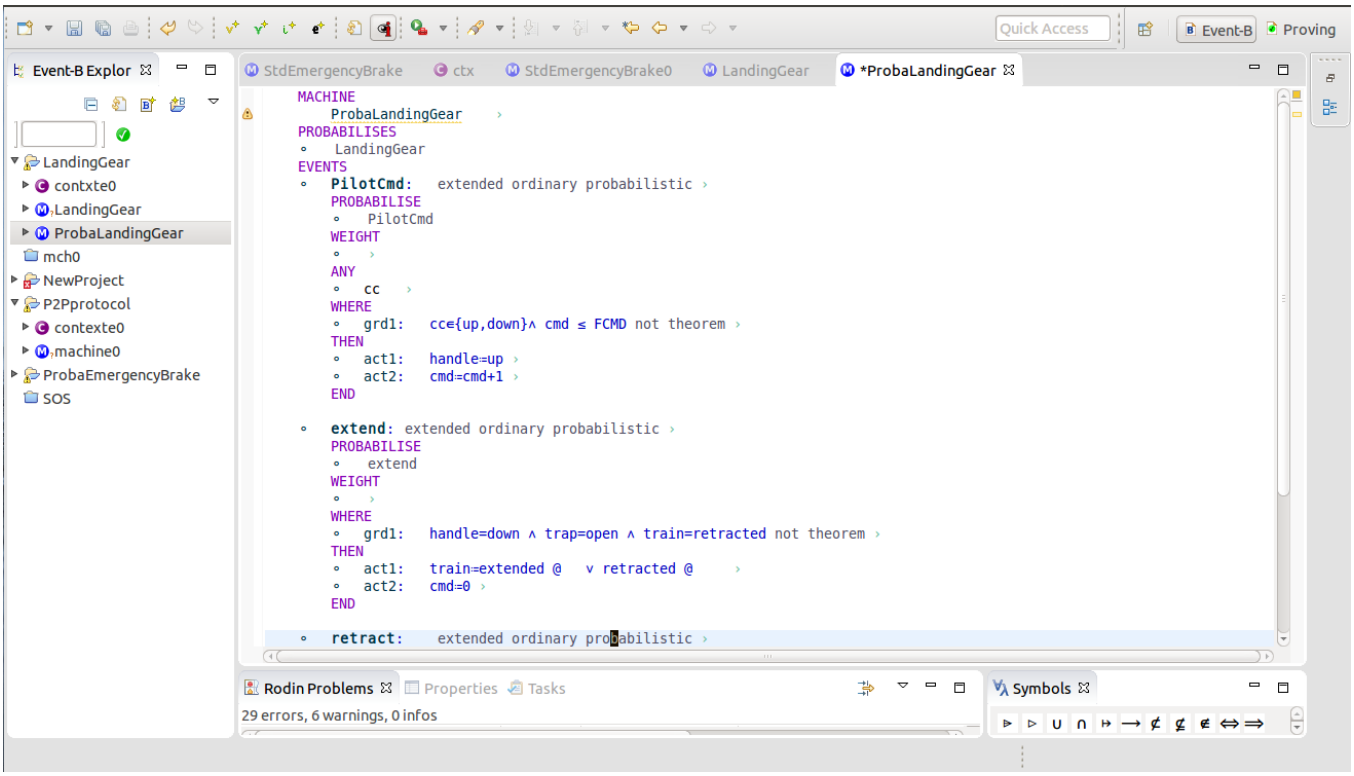

FIGURE 7.12 – Machine B événementiel probabiliste

Écriture des substitutions probabilistes	✓
Annotation des événements par des poids	✓
Génération de la OP (evt/WGHT/NAT)	∼
Génération de la OP (evt/param/pWD)	∼
Génération de la OP (evt/assign/pWD1)	∼
Génération de la OP (evt/assign/pWD2)	∼
Génération de la OP (evt/assign/pWD3)	∼
Génération de la OP (mixedEB/csrt1)	∼
Génération de la OP (machine/pVar)	∼
Génération de la OP (evt/wght/BOUND)	∼
Génération de la OP (evt/param/BOUND)	∼
Génération de la OP (evt/ndpBOUND)	∼
Mise à jour de la OP (evt/assign/pFIS)	∼
Mise à jour de la OP (evt/pINV)	∼
Mise à jour de la OP (machine/pDLF)	∼
Mise à jour de la OP (machine/mDLF)	∼
Réutilisation de la OP (evt/pBOUND)	∼
Réutilisation de la OP (evt/var/pNAT)	∼
Réutilisation de la OP (evt/pVAR)	∼
Intégration du processus de probabilisation	✓
Vérification de la condition (event/param/proba)	✓
Vérification de la condition (evt/EnumSub/proba	✓
Vérification de la condition (evt/PredSub/proba)	✓
Génération d'une machine probabiliste	✓

TABLE 7.1 – Fonctionnalités de l'outil

Pour l'extension de Rodin permettant d'appliquer la probabilisation sur une machine standard, nous avons étendu le noyau de Rodin, les librairies B événementiel principalement les librairies B événementiel AST, tous les composants du noyau B événementiel (l'analyseur de syntaxe, générateur d'obligations de preuve, gestionnaire d'obligations de preuve) ainsi que les interfaces de modélisation et de preuve.

Avancement de l'implémentation. À l'heure actuelle, notre implémentation ne prend pas en compte tous les éléments de l'extension. Nous présentons dans le tableau 7.1 un récapitulatif des fonctionnalités assurées par l'implémentation. ✓ dénote une fonctionnalité supportée alors que ∼ dénote une fonctionnalité en cours d'implémentation. La principale lacune concerne la gestion ainsi que la génération des obligations de preuve. En effet, nous avons implémenté toutes les obligations de preuve, mais nous n'arrivons pas à les faire générer dans l'outil. Le problème est principalement technique, il concerne la connexion de notre implémentation avec les prouveurs de Rodin.

8

Conclusion

L'objectif principal des travaux de recherche présentés dans ce document est l'intégration du raisonnement probabiliste dans la méthode B événementiel, ce qui permet la modélisation ainsi que la vérification d'une large classe d'aspects probabilistes des systèmes.

La problématique d'intégration des probabilités au sein de la méthode B événementiel a été étudiée initialement par Abrial et ses coauteurs dans [78]. Ils suggèrent d'introduire les probabilités comme raffinement du non-déterminisme. En particulier, ils mentionnent que chaque extension au B événementiel pour supporter des probabilités doit respecter certaines exigences :

(1) Le B événementiel doit rester simple et compréhensible,

(2) L'extension doit être générique permettant la description d'une large classe de systèmes.

Nous nous sommes basés sur les consignes énoncées dans ce travail et nous avons proposé une extension probabiliste au B événementiel où nous avons autorisé le remplacement de toutes les sources de non-déterminisme par des probabilités.

Dans un premier temps, nous avons étudié des modèles B événementiel purement probabilistes contenant des probabilités aux différents lieux de non-déterminisme. Pour le non-déterminisme entre événements activables simultanément, nous avons proposé d'annoter chaque événement par un poids. Nous avons proposé de remplacer le choix non-déterministe pour les valeurs des paramètres d'un événement par un choix uniforme discret et nous avons également proposé de remplacer les substitutions non-déterministes par des substitutions probabilistes. De nouveaux éléments de syntaxe ont ainsi été proposés. Par la suite, nous avons étudié la cohérence de ces modèles et nous avons proposé de nouvelles obligations de preuve dédiées aux nouveaux éléments introduits ainsi que des adaptations des obligations de preuve standard pour qu'elles puissent être utilisées dans notre extension purement probabiliste au B événementiel. Nous avons aussi étudié la sémantique des modèles B événementiel purement probabilistes, exprimée en termes de chaînes de Markov discrètes.

Dans un second temps, nous avons étudié des modèles B événementiel contenant à la fois du non-déterminisme et des probabilités. Dans ce cas, on peut conserver du non-déterminisme à certains endroits et le remplacer par des probabilités à d'autres, ce qui permet la description d'une large classe de systèmes. Nous avons ainsi étudié ce type de modèles et nous l'avons désigné par les

modèles B événementiel mixtes. Nous avons étudié la cohérence de ces modèles et leur sémantique en l'exprimant en termes d'automates probabilistes.

Le processus de développement en B événementiel étant basé sur le raffinement, nous avons proposé des approches de développement formel qui permettent l'introduction progressive des probabilités dans un modèle B événementiel par raffinement. Dans une première étape, nous avons proposé une nouvelle approche, *la probabilisation*, qui permet de passer d'un modèle non-déterministe vers un modèle purement probabiliste. Dans une deuxième étape, nous avons étudié une variante spécifique du raffinement : l'ajout de nouveaux événements probabilistes. Nous avons étudié la propriété de convergence presque certaine d'un ensemble d'événements probabilistes en B événementiel et formalisé les conditions nécessaires permettant de démontrer cette propriété par un ensemble d'obligations de preuve.

Afin d'illustrer les éléments de notre extension, nous avons présenté trois études de cas. La première concerne un protocole de communication pair à pair présenté comme un fil conducteur tout au long de ce manuscrit. Les deux autres concernent le train d'atterrisage d'un avion et un système de freinage. Nous avons aussi commencé le développement d'une extension à la plateforme Rodin permettant de mettre en pratique les éléments de l'extension.

Nous notons que plusieurs autres travaux de recherche ont traité l'intégration de probabilités dans B événementiel [54, 58, 98, 53, 91, 92, 93]. Ces travaux ont seulement remplacé les substitutions non-déterministes par des substitutions probabilistes : ils gardent ainsi du non-déterminisme qui apparaît dans le choix entre événements activables simultanément et dans le choix des valeurs des paramètres d'un événement. Ceci rend difficile la modélisation des systèmes purement probabiliste. Il est en effet alors nécessaire d'effectuer des modifications et des réecritures fastidieuses sur les modèles afin d'assurer qu'aucun aspect non-déterministe n'est encore présent, ce qui complique la tâche de modélisation et donne des modèles B événementiel moins lisibles et moins compréhensibles.

Notre extension, qui remplace tous les choix non-déterministes par des probabilités, facilite donc la description des systèmes purement probabilistes et rend leur écriture plus simple et plus intuitive. Nous déduisons ainsi que la première exigence d'Abrial par rapport à la proposition d'extension probabiliste au B événementiel (1) est bien respectée par notre extension. Notre proposition de modèles mixtes contenant à la fois du non-déterminisme et des probabilités permet la description d'une large classe de systèmes, ce qui respecte bien la deuxième exigence de [78] (2).

Perspectives

Comme perspectives à nos travaux de recherche, nous pensons que quatre principales idées peuvent être réalisées :

- La première idée consiste à étudier l'expression ainsi que la vérification des prédicats probabilistes en B événementiel probabiliste. Dans la littérature, plusieurs travaux de recherche [42, 45, 45, 64, 69] ont étudié l'expression et la vérification des prédicats probabilistes. Ces prédicats peuvent être classés en deux catégories : les prédicats probabilistes exprimant la probabilité d'occurence d'un prédicat déterministe [45] ou les prédicats probabilistes qui donnent des expressions réelles sur les variables du modèle [42]. L'intérêt est d'étendre l'ensemble des propriétés exprimables en B événementiel et de raisonner sur de nouveaux types de propriétés qui n'a pas, jusqu'ici, été pris en compte en B événementiel. Une idée pour cela est de formaliser des obligations de preuve pour vérifier ces prédicats. La difficulté concerne l'écriture des ces obligations de preuve ainsi que l'automatisation de leur preuve.

- Le B événementiel dans sa version standard permet la vérification de propriétés exactes de manière binaire, c'est-à-dire des propriétés qui expriment des certitudes et qui sont généralement exprimées comme des invariants. Puisque nous introduisons des probabilités dans le B événementiel, nous nous intéressons à la vérification de propriétés quantitatives exprimées comme des invariants probabilistes. Ce type d'invariants a été étudié dans plusieurs travaux de recherche [57, 79], il a été aussi étudié dans l'extension probabiliste à la méthode B proposée par Hoang [59]. Ces invariants peuvent être considérés comme un cas particulier des prédicats probabilistes. Ils sont généralement donnés par des expressions numériques exprimant des contraintes qui doivent être vérifiées par les variables du modèle. L'intérêt réside dans l'introduction d'un raisonnement quantitatif qui n'a pas été traité auparavant en B événementiel et dans l'expression de contraintes quantitatives sur des modèles B événementiel. Cependant, ce type d'invariants nécessite un raisonnement en termes de nombre réels et non uniquement en termes de booléens. La difficulté concerne ainsi le passage en B événementiel du monde de raisonnement en booléen vers le monde de raisonnement en réels. Comme mentionné déjà dans ce manuscrit, les réels ne sont pas nativement gérés en B événementiel. Pour les rationnels, il est possible de les exprimer en termes d'entiers. Pour les réels, c'est tout de même plus complexe et cela demande des prouveurs et des tactiques de preuves dédiés.

- Les propriétés gérées nativement en B événementiel sont l'invariance, l'absence de blocage et la convergence. Des travaux de recherche ont traité l'expression et la vérification d'autres propriétés probabilistes telles que la fiablité [91, 96], la réactivité [38, 94] ou la convergence presque certaine [54]. Dans notre extension probabiliste, nous avons étudié la propriété de convergence presque certaine d'un ensemble d'événements. La perspective est ici d'étudier l'expression et la vérification de certaines propriétés probabilistes dans notre extension probabiliste. Les travaux ayant traité l'expression et la vérification de la fiabilité et la réactivité n'ont pas fourni de solutions complète pour gérer ces propriétés en B événementiel. En effet, ils considèrent des modèles B événementiel contenant des probabilités (au niveau de substitutions) et les basculent vers des *model checkers* (principalement Prism [70]) pour vérifier la fiabilité ou la réactivité. Ainsi leur travail n'offre pas de moyen de vérifier ces propriétés en B événementiel par des techniques de preuve ; L'intérêt ainsi est de proposer une solution complète permettant à la fois l'expression et la vérification par des obligations de preuve dédiées ces propriétés ou d'autres dans le B événementiel probabiliste.

- Le processus de développement en B événementiel est basé sur le raffinement. Dans notre travail, nous avons proposé un processus de développement basé sur le raffinement pour introduire des probabilités dans un modèle B événementiel, mais ce processus reste incomplet. Nous n'avons en effet traité que l'ajout de nouveaux événements probabilistes dans une machine B événementiel ou la probabilisation d'un ensemble d'événements non-déterministes vers un ensemble d'événements probabilistes. L'ajout de nouvelles variables ainsi que le raffinement des données n'ont pas été traité dans notre travail. Ainsi, plusieurs voies de recherche traitant le raffinement probabiliste dans le B événementiel probabiliste sont possibles : La première voie consiste à étudier le raffinement des données dans le B événementiel probabiliste. L'idée est d'étudier l'ajout ou la suppresion de variables d'une machine B événementiel probabiliste lors d'une étape de raffinement. Nous n'avons pas non plus traité le cas du raffinement direct d'un événement probabiliste par un autre, ou les cas spécifiques de division d'un événement probabiliste ou de regroupement d'événements probabilistes. Une deuxième voie de recherche consiste ainsi à étudier le raffinement d'un événement probabiliste par un autre. La difficulté est de définir la relation entre les poids des deux événements abstrait et concret. L'idée consiste aussi à étudier le raffinement d'une substitution probabiliste énumérée ou qualitative. Pour cela, il s'agit d'étudier l'adaptation des obligations de preuve de raffinement standards à ce contexte et de dégager les nouvelles obligations de preuve servant à assurer la

validité de ce raffinement. Pour le raffinement des substitutions probabilistes, il est possible de s'inspirer des travaux de Morgan sur les substitutions probabilistes présentées dans [76].

Une troisième voie de recherche consiste à étudier les cas spécifiques de décomposition (*split*) ou de regroupement (*merge*) pour les événements probabilistes. Principalement, la difficulté concerne aussi la relation entre les valeurs de probabilité des événements abstraits et des événements concrets. L'intérêt global ici est de proposer un processus de raffinement probabiliste complet qui permet autant de flexibilité dans le cadre probabiliste que dans le cadre standard.

Nous notons aussi que notre extension à la plateforme Rodin n'est pas complète. En particulier, la partie sur la génération des obligations de preuve n'est pas finalisée. La difficulté concerne principalement l'implémentation de certaines obligations de preuve ainsi que la connexion de certains prouveurs avec Rodin pour gérer ces obligations. Une continuité de notre travail consiste à poursuivre le développement de l'extension probabiliste à Rodin, l'objectif étant de finir l'intégration de la partie sur la génération des obligations de preuve.

Bibliographie

[1] Atelierb. http://www.atelierb.eu/. Accessed : 2017-02-23. 18

[2] B2latex. http://wiki.event-b.org/index.php/B2Latex. Accessed : 2017-02-22. 125

[3] B4free. http://www.b4free.com/. Accessed : 2017-02-23. 18

[4] Bittorent. http://www.bittorrent.com/lang/fr/. Accessed : 2017-02-27. 28

[5] Clearsy. http://www.clearsy.com/. Accessed : 2017-02-23. 18

[6] Deployproject. http://www.deploy-project.eu/. Accessed : 2017-04-14. 18, 117, 125

[7] Eclipse. http://www.eclipse.org. Accessed : 2017-02-22. 125

[8] Peer-to-peer. http://www.prismmodelchecker.org/. Accessed : 2017-02-27. 28

[9] Prob. http://wiki.event-b.org/index.php/ProB. Accessed : 2017-02-22. 18, 125

[10] Rodin platform. http://www.event-b.org. Accessed : 2017-02-22. 18, 125

[11] Rodin project. http://rodin.cs.ncl.ac.uk. Accessed : 2017-02-22. 18, 125

[12] Uml-b. http://wiki.event-b.org/index.php/UML-B. Accessed : 2017-02-22. 125

[13] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2) :253–284, 1991. 24

[14] J. Abrial. Tools for constructing large systems (a proposal). *Rigorous Development of Complex Fault-Tolerant Systems. M. Butler, etc.(Eds). LNCS*, 4157. 125

[15] J.-R. Abrial. Extending b without changing it (for developing distributed systems). In *1st Conference on the B method*, volume 11, 1996. 17

[16] J.-R. Abrial. *The B-Book : Assigning programs to meanings*. Cambridge University Press, 2005. 10, 17, 24

[17] J.-R. Abrial. *Modeling in Event-B : system and software engineering*. Cambridge University Press, 2010. 10, 12, 17, 18, 21, 22

[18] J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin. Rodin : an open toolset for modelling and reasoning in event-b. *International journal on software tools for technology transfer*, 12(6) :447–466, 2010. 12

[19] J.-R. Abrial, M. Butler, S. Hallerstede, and L. Voisin. An open extensible tool environment for event-b. In *International Conference on Formal Engineering Methods*, pages 588–605. Springer, 2006. 125

[20] J.-R. Abrial, D. Cansell, and D. Méry. A mechanically proved and incremental development of ieee 1394 tree identify protocol. *Formal aspects of computing*, 14(3) :215–227, 2003. 17

[21] J.-R. Abrial and L. Mussat. Introducing dynamic constraints in b. In *International Conference of B Users*, pages 83–128. Springer, 1998. 17

[22] P. Audebaud and C. Paulin-Mohring. Proofs of randomized algorithms in coq. *Science of Computer Programming*, 74(8) :568–589, 2009. 10, 11

[23] R. Back. *On the correctness of refinement in program development*. PhD thesis, Ph. D. thesis, Report A-1978-4, Department of Computer Science, University of Helsinki, 1978. 24

[24] R.-J. Back. A calculus of refinements for program derivations. *Acta Informatica*, 25(6) :593–624, 1988. 24

[25] R.-J. R. Back and R. Kurki-Suonio. Decentralization of process nets with centralized control. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 131–142. ACM, 1983. 10

[26] R.-J. R. Back and J. von Wright. Duality in specification languages : a lattice-theoretical approach. *Acta Informatica*, 27(7) :583–625, 1990. 24

[27] C. Baier. On algorithmic verification methods for probabilistic systems. *Universität Mannheim*, 1998. 35

[28] C. Baier, J.-P. Katoen, et al. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008. 11, 33, 35, 36, 107

[29] G. Barthe, C. Fournet, B. Grégoire, P.-Y. Strub, N. Swamy, and S. Zanella-Béguelin. Probabilistic relational verification for cryptographic implementations. In *ACM SIGPLAN Notices*, volume 49, pages 193–205. ACM, 2014. 10, 11

[30] M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta informatica*, 20(3) :207–226, 1983. 11

[31] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and software verification : model-checking techniques and tools*. Springer Science & Business Media, 2013. 11

[32] D. Bert and F. Cave. Construction of finite labelled transition systems from b abstract systems. In *Integrated Formal Methods*, volume 1945 of *LNCS*, pages 235–254. Springer, 2000. 37

[33] A. Bianco and L. De Alfaro. Model checking of probabilistic and nondeterministic systems. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 499–513. Springer, 1995. 11

[34] I. Board. Ariane 5–flight 501 failure. *The Chairman of the Board : Prof. JL LIONS, available at http ://www. di. unito. it/˜ damiani/ariane5rep. html*, 1996. 9

[35] F. Boniol and V. Wiels. The landing gear system case study. In *ABZ 2014 : The Landing Gear Case Study*, pages 1–18. Springer, 2014. 111

[36] M. Butler and S. Hallerstede. The rodin formal modelling tool. In *Electronic Workshop In Computing*, 2007. 18, 125

[37] M. Butler and I. Maamria. Practical theory extension in event-b. In *Theories of Programming and Formal Methods*, pages 67–81. Springer, 2013. 50

[38] W. W. Chu and C.-M. Sit. Estimating task response time with contentions for real-time distributed systems. In *Real-Time Systems Symposium, 1988., Proceedings.*, pages 272–281. IEEE, 1988. 10, 41, 137

[39] E. M. Clarke and I. A. Draghicescu. Expressibility results for linear-time and branching-time logics. In *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, pages 428–437. Springer, 1988. 11

[40] E. M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT press, 1999. 11

[41] E. M. Clarke, W. Klieber, M. Nováček, and P. Zuliani. Model checking and the state explosion problem. In *LASER Summer School on Software Engineering*, pages 1–30. Springer, 2011. 11

[42] M. A. Colón, S. Sankaranarayanan, and H. B. Sipma. Linear invariant generation using non-linear constraint solving. In *Computer Aided Verification*, pages 420–432. Springer, 2003. 136

[43] P. M. Defense. Software problem led to system failure at dhahran, saudi arabia. *US GAO Reports, report no. GAO/IMTEC-92-26*, 1992. 9

[44] C. Dehnert, D. Gebler, M. Volpato, and D. N. Jansen. *On Abstraction of Probabilistic Systems*, pages 87–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. 11

[45] J. Den Hartog and E. P. de Vink. Verifying probabilistic programs using a hoare like logic. *International Journal of Foundations of Computer Science*, 13(03) :315–340, 2002. 136

[46] E. W. Dijkstra, E. W. Dijkstra, E. W. Dijkstra, E.-U. Informaticien, and E. W. Dijkstra. *A discipline of programming*, volume 1. prentice-hall Englewood Cliffs, 1976. 18

[47] J. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, and M. Sighireanu. Cadp a protocol validation and verification toolbox. In *Computer Aided Verification*, pages 437–440. Springer, 1996. 11

[48] O. Goldreich. Probabilistic proof systems. In *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, pages 39–72. Springer, 1999. 10, 11

[49] M. J. Gordon and T. F. Melham. Introduction to hol a theorem proving environment for higher order logic. 1993. 11

[50] H. Haghighi. Pz : A z-based formalism for modeling probabilistic behavior. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 4(1) :29–37, 2010. 10

[51] H. Haghighi and M. Afshar. A z-based formalism to specify markov chains. *Computer Science and Engineering*, 2(3) :24–31, 2012. 10, 11

[52] S. Hallerstede and M. Butler. Performance analysis of probabilistic action systems. *Formal Aspects of Computing*, 16(4) :313–331, 2004. 10

[53] S. Hallerstede and T. S. Hoang. Qualitative reasoning for the dining philosophers—extended abstract—. 11, 12, 40, 43, 44, 67, 136

[54] S. Hallerstede and T. S. Hoang. Qualitative probabilistic modelling in event-b. In *Integrated Formal Methods*, pages 293–312. Springer, 2007. 11, 12, 39, 40, 89, 90, 100, 136, 137

[55] D. Harel. Statecharts : A visual formalism for complex systems. *Science of computer programming*, 8(3) :231–274, 1987. 10

[56] D. Harel and M. Politi. *Modeling reactive systems with statecharts : the STATEMATE approach*. McGraw-Hill, Inc., 1998. 10

[57] T. S. Hoang. *The development of a probabilistic B-method and a supporting toolkit*. PhD thesis, The University of New South Wales, 2005. 10, 11, 89, 137

[58] T. S. Hoang. Reasoning about almost-certain convergence properties using event-b. *Science of Computer Programming*, 81 :108–121, 2014. 11, 12, 40, 43, 44, 67, 89, 90, 136

[59] T. S. Hoang, Z. Jin, K. Robinson, A. McIver, and C. Morgan. Probabilistic invariants for probabilistic machines. In *ZB 2003 : Formal Specification and Development in Z and B*, pages 240–259. Springer, 2003. 11, 137

[60] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10) :576–580, 1969. 11

[61] G. J. Holzmann. The model checker spin. *IEEE Transactions on software engineering*, 23(5) :279–295, 1997. 11

[62] G. Huet, G. Kahn, and C. Paulin-Mohring. The coq proof assistant a tutorial. *The Coq Project*, 2004. 11

[63] J. Hurd. *Formal verification of probabilistic algorithms*. PhD thesis, University of Cambridge, Computer Laboratory, 2003. 10, 11

[64] J. Hurd, A. McIver, and C. Morgan. Probabilistic guarded commands mechanized in hol. *Electronic Notes in Theoretical Computer Science*, 112 :95–111, 2005. 10, 11, 136

[65] M. Huth and M. Ryan. *Logic in Computer Science : Modelling and reasoning about systems*. Cambridge university press, 2004. 11

[66] C. B. Jones. *Systematic software development using VDM*, volume 2. Prentice-Hall Englewood Cliffs, NJ, 1986. 10

[67] B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *Logic in Computer Science, 1991. LICS'91., Proceedings of Sixth Annual IEEE Symposium on*, pages 266–277. IEEE, 1991. 11

[68] J.-P. Katoen. *Abstraction of Probabilistic Systems*, pages 1–3. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. 11

[69] J.-P. Katoen, A. K. McIver, L. A. Meinicke, and C. C. Morgan. Linear-invariant generation for probabilistic programs. In *Static Analysis*, pages 390–406. Springer, 2010. 136

[70] M. Kwiatkowska, G. Norman, and D. Parker. Prism : Probabilistic symbolic model checker. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 200–204. Springer, 2002. 35, 41, 137

[71] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(3) :872–923, 1994. 10

[72] K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1) :134–152, 1997. 11

[73] M. Leuschel and M. Butler. Prob : an automated analysis toolset for the b method. *International Journal on Software Tools for Technology Transfer*, 10(2) :185–203, 2008. 125

[74] R. Mayr, N. B. Henda, and P. A. Abdulla. Decisive markov chains. *Logical Methods in Computer Science*, 3, 2007. 96, 97

[75] A. McIver, C. Morgan, and T. Hoang. Probabilistic termination in b. In D. Bert, J. Bowen, S. King, and M. Waldén, editors, *ZB 2003 : Formal Specification and Development in Z and B*, volume 2651 of *Lecture Notes in Computer Science*, pages 216–239. Springer Berlin Heidelberg, 2003. 89

[76] A. McIver and C. C. Morgan. *Abstraction, refinement and proof for probabilistic systems*. Springer Science & Business Media, 2006. 138

[77] C. Morgan. *Programming from specifications*. Prentice Hall,, 1994. 24

[78] C. Morgan, T. S. Hoang, and J.-R. Abrial. The challenge of probabilistic event b—extended abstract—. In *ZB 2005 : Formal Specification and Development in Z and B*, pages 162–171. Springer, 2005. 11, 12, 38, 43, 44, 67, 81, 135, 136

[79] C. Morgan and A. McIver. pgcl : formal reasoning for random algorithms. *South African Computer Journal*, pages 14–27, 1999. 137

[80] J. M. Morris. A theoretical basis for stepwise refinement and the programming calculus. *Science of Computer programming*, 9(3) :287–306, 1987. 24

[81] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. Srivas. Pvs : Combining specification, proof checking, and model checking. In *International Conference on Computer Aided Verification*, pages 411–414. Springer, 1996. 11

[82] L. C. Paulson. *Isabelle : A generic theorem prover*, volume 828. Springer Science & Business Media, 1994. 11

[83] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *International Colloquium on Automata, Languages, and Programming*, pages 15–32. Springer, 1985. 11

[84] M. L. Puterman. *Markov decision processes : discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 11

[85] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2) :250–273, 1995. 36

[86] K. Sere and E. Troubitsyna. Probabilities in action systems. In *Proc. of the 8th Nordic Workshop on Programming Theory*, pages 373–387, 1996. 10, 11

[87] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM (JACM)*, 32(3) :733–749, 1985. 11

[88] C. Snook and M. Butler. Uml-b : Formal modeling and design aided by uml. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15(1) :92–122, 2006. 125

[89] J. M. Spivey and J. Abrial. *The Z notation*. Prentice Hall Hemel Hempstead, 1992. 10

[90] M. Stoelinga. An introduction to probabilistic automata. *Bulletin of the EATCS*, 78(176-198) :2, 2002. 11

[91] A. Tarasyuk, E. Troubitsyna, and L. Laibinis. Reliability assessment in event-b development. *NODES 09*, page 11, 2009. 11, 12, 40, 43, 44, 57, 67, 136, 137

[92] A. Tarasyuk, E. Troubitsyna, and L. Laibinis. Towards probabilistic modelling in event-b. In *Integrated Formal Methods*, pages 275–289. Springer, 2010. 11, 12, 40, 41, 43, 44, 67, 136

[93] A. Tarasyuk, E. Troubitsyna, and L. Laibinis. Integrating stochastic reasoning into event-b development. *Formal Aspects of Computing*, 27(1) :53–77, 2015. 11, 12, 40, 41, 43, 44, 57, 67, 136

[94] K. S. Trivedi, S. Ramani, and R. Fricks. Recent advances in modeling response-time distributions in real-time systems. *Proceedings of the IEEE*, 91(7) :1023–1037, 2003. 10, 41, 137

[95] R. J. VanGlabbeek, S. A. Smolka, and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Information and Computation*, 121(1) :59–80, 1995. 35

[96] A. Villemeur. *Reliability, Availability, Maintainability and Safety Assessment, Assessment, Hardware, Software and Human Factors*, volume 2. Wiley, 1992. 10, 41, 137

[97] E. Yilmaz. *Tool support for qualitative reasoning in Event-B*. PhD thesis, Master Thesis ETH Zürich, 2010, 2010. 40

[98] E. Yilmaz and T. S. Hoang. Development of rabin's choice coordination algorithm in event-b. *Electronic Communications of the EASST*, 35, 2011. 11, 12, 40, 43, 44, 67, 136

[99] S. Yovine. Kronos : A verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1) :123–133, 1997. 11

Liste des tableaux

7.1 Fonctionnalités de l'outil 134

Table des figures

1.1 Comment introduire des probabilités au sein du B événementiel ? 13

2.1 Structure d'un contexte 19
2.2 Structure d'une machine 20
2.3 Formes des événements en B événementiel 20
2.4 Structure d'un événement abstrait et d'un événement concret 25
2.5 Principe général du protocole 28
2.6 Stratégie du développement du protocole en B événementiel 29
2.7 Première machine 30
2.8 Deuxième contexte 30
2.9 Deuxième machine 31
2.10 Troisième machine 34
2.11 Exemple d'un système de transitions 35
2.12 Exemple d'un automate probabiliste 36
2.13 Exemple d'une chaîne de Markov discrète 37
2.14 Extrait du système de transitions correspondant à la machine $P2P_2$, avec N=2 et K=2 38

3.1 Machine Mch 44
3.2 Sémantique opérationnelle de la machine Mch 45
3.3 Construction détaillée de la sémantique opérationnelle de la machine Mch 45
3.4 Choix probabiliste de l'événement à exécuter 46
3.5 Choix uniforme de la valeur du paramètre t 47
3.6 Construction détaillée de la sémantique de la machine PMch 47
3.7 Chaîne de Markov résultante 48
3.8 Forme d'un événement probabiliste 51
3.9 Structure d'une machine probabiliste 51
3.10 Une description probabiliste du protocole P2P en B événementiel probabiliste . . . 52
3.11 Extrait de la construction détaillée de la chaîne de Markov discrète du protocole P2P avec N=2 and K=2 63
3.12 Extrait de la chaîne de Markov du protocole P2P, avec N=2 et K=2 65

4.1 Structure d'une machine mixte 68
4.2 Formes des événements en B événementiel mixte 68
4.3 Une machine B événementiel mixte partielle 73
4.4 Construction détaillée de la sémantique de la machine Exemple1 73
4.5 Sémantique de la machine Exemple1 74
4.6 Une machine B événementiel mixte 75
4.7 Sémantique de la machine Exemple2 75
4.8 Une description du protocole P2P en B événementiel mixte 77
4.9 Extrait de l'automate probabiliste du protocole P2P avec N=2 and K=2 (figure à modifier) 78

5.1 Comment introduire des probabilités au sein du B événementiel ? . . . 82
5.2 Une machine B événementiel mixte du protocole P2P . . . 86
5.3 Une machine B événementiel purement probabiliste du protocole P2P . . . 88
5.4 Machine B événementiel probabiliste M_1 . . . 92
5.5 Extrait de la sémantique opérationnelle de M1 . . . 92
5.6 Machine B événementiel probabiliste M_2 . . . 93
5.7 Extrait de la sémantique opérationnelle de M_2 . . . 94
5.8 Processus de développement du protocole P2P . . . 101
5.9 Une machine B événementiel mixte du protocole P2P . . . 102
5.10 Machine $P2P_1$' . . . 103
5.11 Machine $P2P_1$'' . . . 104
5.12 Machine $P2P_M$ . . . 106

6.1 Principe du fonctionnement du train d'atterrissage . . . 112
6.2 Stratégie de développement du système du train d'atterrissage en B événementiel . . . 112
6.3 Contexte initial . . . 113
6.4 Première machine . . . 114
6.5 Deuxième machine . . . 117
6.6 Stratégie de développement du système de freinage en B événementiel . . . 118
6.7 Contexte initial . . . 119
6.8 Première machine . . . 119
6.9 Deuxième machine . . . 120
6.10 Troisième machine . . . 122
6.11 Quatrième machine . . . 123
6.12 Machine probabiliste . . . 124

7.1 Architecture de Rodin . . . 126
7.2 Interaction entre les composants du noyau B événementiel . . . 127
7.3 Interface de modélisation dans l'outil Rodin . . . 128
7.4 Interface de preuve dans l'outil Rodin . . . 128
7.5 Interaction entre les interfaces de Rodin et le noyau B événementiel . . . 129
7.6 Ajout d'un poids à un événement . . . 130
7.7 Annotation d'un événement par un poids . . . 130
7.8 Ajout d'une substitution probabiliste . . . 131
7.9 Machine non-déterministe du LandingGear . . . 132
7.10 Application du processus de probabilisation . . . 132
7.11 Précision du nom de la machine . . . 133
7.12 Machine B événementiel probabiliste . . . 133

UNIVERSITE BRETAGNE LOIRE

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Mohamed Amine AOUADHI

Introduction de raisonnement probabiliste dans la méthode B événementiel

Introducing probabilistic reasoning within Event-B

Résumé

Les méthodes de modélisation et de vérification formelles à base de preuves, par exemple le B événementiel, ne permettent pas, à ce jour, de bien prendre en compte l'ensemble des aspects quantitatifs des systèmes réels. En particulier, l'ajout d'aspects probabilistes dans les systèmes B événementiel est une problématique qui n'a pas été bien étudiée dans l'état de l'art. La difficulté réside principalement dans l'expression des probabilités ainsi que la vérification des aspects probabilistes dans ce formalisme.

Dans cette thèse, une extension probabiliste au B événementiel est proposée pour permettre la description ainsi que la vérification des aspects probabilistes des systèmes. Nous désignons cette extension par le B événementiel probabiliste. Dans cette extension, nous proposons de remplacer toutes les sources de non-déterminisme en B événementiel par des probabilités, ce qui permettra ainsi la description des comportements purement probabilistes. Le processus de développement en B événementiel étant basé sur le raffinement, nous proposons plusieurs approches de développement basées sur le raffinement qui permettront l'intégration progressive des probabilités. En particulier, nous étudions la convergence presque certaine d'un ensemble d'événements dans cette extension. La méthode B événementiel est équipée de la plateforme Rodin que nous étendons pour permettre la prise en compte des éléments de l'extension.

Les différents aspects de ce travail sont illustrés par plusieurs études de cas : un protocole de communication pair à pair, le système de train d'atterrissage d'un avion et un système de freinage d'urgence.

Mots clés

Modélisation, B événementiel, Probabilités, Raffinement, Méthodes formelles.

Abstract

From the best of our knowledge, proof based modeling and verification methods, for example Event-B, are not capable of handling the complete spectrum of quantitative aspects from real-life systems. In particular, modeling probabilistic aspects of systems within Event-B is a problem which has not been well studied in the state of the art. The main difficulties lie in the expression of probabilities and the verification of probabilistic aspects within Event-B. This thesis presents a probabilistic extension of Event-B to support modeling and verification of probabilistic aspects of systems. We denote this extension by probabilistic Event-B. We propose to replace all the non-deterministic choices by probabilistic choices, which allows the description of fully probabilistic systems. The development process in Event-B is based on refinement, we therefore propose some development approaches based on refinement which permit the progressive integration of probabilities in Event-B models. In particular, we study the almost-certain convergence of a set of events within this extension. The Event-B method is equipped with the Rodin platform which we extend to take into consideration the new added elements of our extension.

The different aspects of this work are illustrated by several case studies: a peer-to-peer communication protocol, a landing gear system and an emergency brake system.

Key Words

Modeling, Event-B, Probabilities, Formal methods, Refinement.