A Formal Treatment of Agents, Goals and Operations Using Alternating-Time Temporal Logic

Christophe Chareton, Julien Brunel, David Chemouil

A FORMAL TREATMENT OF AGENTS, GOALS AND OPERATIONS USING ALTERNATING-TIME TEMPORAL LOGIC

CHRISTOPHE CHARETON, JULIEN BRUNEL, AND DAVID CHEMOUIL
ONERA – THE FRENCH AEROSPACE LAB
F-31055 TOULOUSE, FRANCE

Published in *Lecture Notes in Computer Science 7021*. DOI: 10.1007/978-3-642-25032-3_13.

Abstract. The aim of this paper is to provide a formal framework for Requirements Engineering modelling languages featuring agents, behavioural goals and operations as main concepts. To do so, we define Khi, a core modelling language, as well as its formal semantics in terms of a fragment of the multi-agent temporal logic ATL*, called ATL_{Khi}. Agents in the sense of concrete and provided entities, called actors, are defined by their capabilities. They also pursue behavioural goals that are realised by operations, which are themselves gathered into abstract, required, agents, that we call roles. Then a notion of assignment, between (coalitions of) actors and roles is defined. Verifying the correctness of a given assignment then reduces to the validity of an ATL_{Khi} formula that confronts the capabilities of (coalitions of) actors with the operations in roles played by the said actors. The approach is illustrated through a toy example featuring an online shopping marketplace.

1. Introduction

Requirements Engineering (RE) is that part of software or systems engineering concerned with describing the problem domain and determining requirements for a system to be developed [10]. An important part of the RE discipline is concerned with isolating concepts for RE modelling. In this setting, the aim of this paper is to contribute to the formalisation of RE modelling languages featuring agents, behavioural goals and operations as main concepts.

Goals are a central concept of RE modelling languages. They are characterised as prescriptive statements over a system under study. Most of the time, goals are expressed in natural language. However, some approaches propose a framework to describe classes of goals formally. In particular, Kaos defines *behavioural goals* [10, 12, 13] in terms of a variant of Linear Temporal Logic (LTL) [15].

A goal may be refined into a combination of several sub-goals and, possibly, domain properties which are descriptive statements about the domain environment. Alternative refinements for a given goal can also be expressed.

In Kaos, the process of refining high-level goals into more precise ones ends with the production of two different kinds of goals: requirements and expectations. Requirements (resp. expectations) are under responsibility of agents in the software (resp. the environment). Requirements corresponding to behavioural goals are then realised by *operations*. In other approaches, the concepts of commitments [6] or tasks [18] play an analogous role.

In Kaos, an operation is defined by descriptive *domain pre-* and *post-conditions* that characterise its effects. Furthermore, the execution of an operation is constrained by prescriptive *required pre-*, *post-* and *trigger conditions* (trigger conditions are *sufficient* conditions for the

execution of an operation, while pre-conditions are *necessary* conditions). The three types of required conditions are inherited from the requirements realised by the said operation [10–13].

Now, the concept of *agent* stands for an active component in the system (software or environment). Agents may be related to goals in two ways. A first relation is between goals and the agents that are *responsible* for them. This relation is present in languages such as KAOS, TROPOS [3], *i** [18], ALBERT and ALBERT II [8, 9]. Furthermore, TROPOS and *i** add a second relation between goals and the agents *aiming* for them. This consideration helps at determining why each goal appears in a model.

The concept of agent itself has given rise to different characterisations. First comes a description of the agents that are "provided". In this article, we call *actor* such an agent. Actors have *capabilities*, *i.e.*, ways to influence the evolution of the system. Capabilities are commonly considered in the literature, even in methods that do not distinguish several characterisations of agents, such as KAOS. Actors also pursue their own relative goals. This relation between actors and goals raises a social dimension: actors are described as having intentions and interactions with each others. These relations, sometimes dubbed *distributed intentionality*, are at the core of *i** [17, 18] and are also present in TROPOS [3].

The second characterisation emphasises "required" agents, identified with the set of operations they perform. A relation of assignment between roles and actors makes the latter, through the roles they play, *responsible* for the operations in these roles. For instance, agents in KAOS enjoy a responsibility relation w.r.t. goals, and they also come with capabilities (in the form of monitored and controlled variables). The responsibilities of agents are also central in ALBERT [8] and ALBERT II [9]. The latter characterises them with a set of constrained behavioural goals they are responsible for. TROPOS and affiliated literature propose a concept of role for agents seen as sets of actions executed along the run of the system [3, 16].

So, actors aim for goals. But they are also responsible for goals, through the roles they play. And then a question arises, which we call the *assignment problem*: are the actors in charge of a role able to fulfill it?

Several techniques have been proposed to solve this question, notably through the introduction of commitments between agents [5–7, 14]. In this setting, actors commit themselves to other actors for realising certain actions. The question is whether each actor is able to support its role, which is identified with a set of commitments. It is tackled by formalising agents capabilities and commitments in a *propositional* language. We feel this use of a propositional language is limiting in the purpose of formalising the relations between roles and actors' capabilities. Indeed, it does not treat either the precedence relations between operations in the system, which determine the interactions between actors, nor the very existence of these actors and their effective actions upon the system. Actors are only taken into account and described in the informal part of this language.

Thus, some methods in RE offer means for describing precedence among operations in the system and others for describing interacting, social actors. We think that there is a strong interest in combining both approaches in a single language with a semantics including the succession between operations, actors' capabilities and their interactions.

In this paper, we define KHI, a core modelling language, to deal with the assignment problem while retaining the behavioural semantics present in many propositions, especially in KAOS. The formal semantics of KHI is expressed in terms of a multi-agent temporal logic $\mathrm{ATL}_{\mathrm{KHI}}$, a fragment of the decidable logic ATL*. Verifying the correctness of a given assignment then reduces to the satisfaction of an $\mathrm{ATL}_{\mathrm{KHI}}$ formula.

To the best of our knowledge, this is the first proposition that reconciles goal- and agent-oriented RE languages in a formal framework with precedence and multi-agent expression.

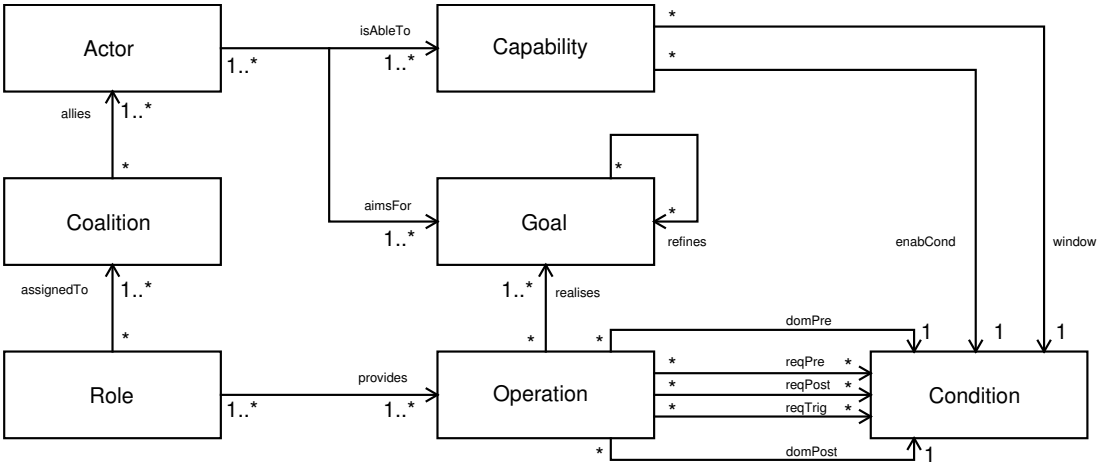

FIGURE 1. Metamodel of language KHI

This paper is organised as follows. In Sect. 2, we present the metamodel of the KHI language, illustrated with a toy example featuring an online shopping marketplace. In Sect. 3, the semantics of KHI is given in terms of ATL_{KHI} formulas. In particular, the assignment problem is formalised. In Sect. 18, we discuss our framework and elaborate on our future work in Sect. 19.

2. THE KHI LANGUAGE

This section gives and comments the different elements and relations in language KHI.

2.1. **The Metamodel.** We first give a brief overview of the metamodel for language KHI (see Fig. 1) before illustrating its concepts in more details on our toy example.

Actors are available agents. They aim for goals that are structured through refinements. Goals may not all be described formally but this work only considers those goals that are formalisable using LTL. Leaf goals of this sort are then realised into operations. Operations are identified with their effect, described using domain pre- and post-conditions. Their scope of application is also constrained by required pre-, post- and trigger conditions. Then, operations are gathered into roles that may be seen as specifications of "required" agents.

Actors also have capabilities. Each capability is described with a pre-condition and a *window* characterising the action it enables (and explained in Sect. 2.3).

Finally, actors can be gathered into coalitions, which are then assigned roles. The effective ability of a coalition to play the roles it is assigned is determined by the capabilities of actors it gathers. Considering coalitions enables to consider agents interactions and to express that many actors may cooperate to play a role.

Note that we do not deal with domain properties or OR-refinements in this paper as it is not needed for the presentation of our formal framework. This could be added later and would be dealt with in a similar way as it is in KAOS.

2.2. **Actors and Goals.** We give in Fig. 2 the goal model in language KHI for our case study. It concerns an online shopping website and actors interacting with it. As in TROPOS, actors are represented as dashed ellipses labelled in circles. An ellipse contains the set of goals the enclosing actor aims for. Goals are represented by trapezoids. They are progressively refined into sub-goals. The graphical representation for goals and their refinements is the same as in KAOS.

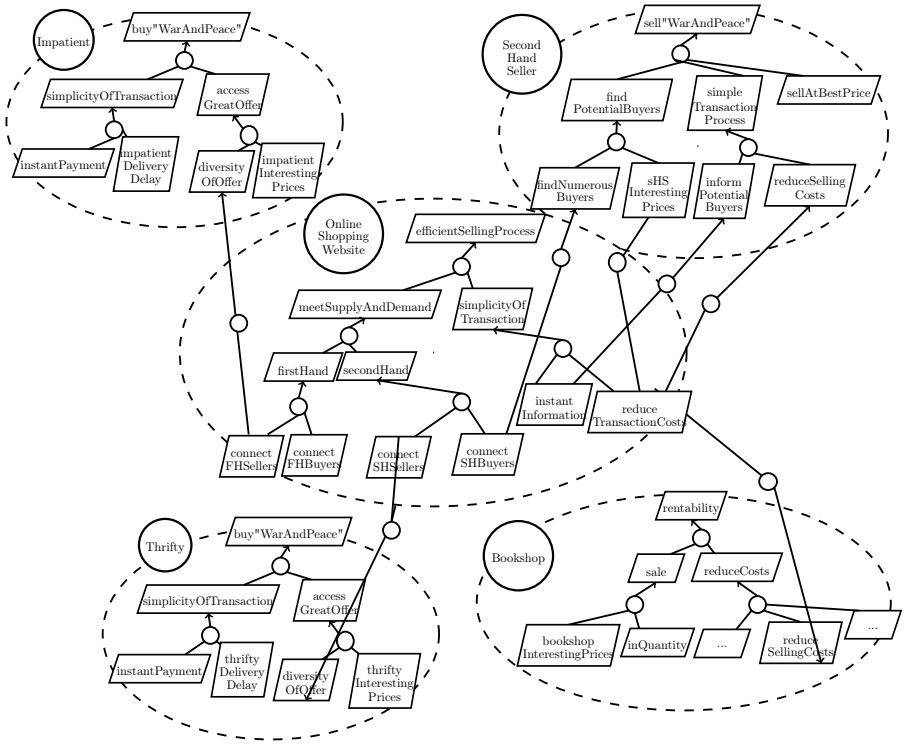

FIGURE 2. Actors and goals for the online shopping marketplace

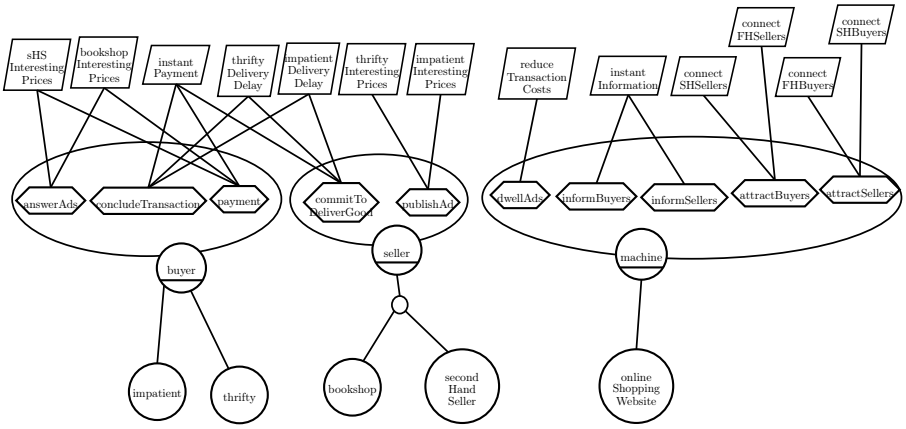

FIGURE 3. Leaf goals, operations, roles and assignments for the online shopping marketplace

For instance, the goal *efficientSeelingProcess* for the website is refined into sub-goals *meetSupplyAndDemand* and *simplicityOfTransaction*.

There are two potential buyers, a *thrifty* one and an *impatient* one. Each of them holds its own goal model. They both want to buy the novel *War and Peace*. Their goal models only differ by the specifications issued from their respective goals *deliveryDelay* and *interestingPrices*, given

commitToDeliverGood	$domPre := \top$ ("true") $domPost := cD < 25$ $reqTrig\ for\ instantPayment := s.o = b.o$ $reqPost\ for\ thriftyDeliveryDelay := cD < 12$ $reqPost\ for\ impatientDeliveryDelay := cD < 7$
publishAd	$domPre := \top$ $domPost := s.o < 30$ $reqTrig\ for\ thriftyInterestingPrices := \top$ $reqPost\ for\ thriftyInterestingPrices := s.o < 12$ $reqPost\ for\ impatientInterestingPrices := s.o < 20$

TABLE 1. Specifications of operations in role *seller*

bookshop	*proposePrice*	$enabCond := \top$ $window := s.o \in [15, 30]$
	expedition	$enabCond := s.o = b.o$ $window := cD \in [4, 25]$
secondHandSeller	*proposePrice*	$enabCond := \top$ $window := s.o \in [9, 20])$
	expedition	$enabCond := s.o = b.o$ $window := cD \in [11, 25]$

TABLE 2. Capabilities of the bookshop and the second hand seller

in Table 1. The *impatient* wants the novels to be delivered within 7 days but may pay for it up to 20 €; while *thrifty* accepts to wait up to 12 days but is unwilling to pay more than 12 € for it. Two different sellers, a bookshop and a second hand seller, having their own goals to satisfy, propose their goods.

The leaf goals in a model are realised by operations to perform. Operations appear in hexagons in Fig. 3. The concept of operations used in KHI follows the definition given for operations in KAOS: they are identified each by a *domPre* and a *domPost* condition. And their executions are constrained by *reqPre*, *reqPost* and *reqTrig* conditions. Thus, as appears in Table 1, *committedDelay<12* is a *reqPost* for goal *thriftyDeliveryDelay* and *committedDelay<7* is a *reqPost* for goal *impatientDeliveryDelay*. The action of giving such specifications of operations in order to realise goals is called *operationalisation*.

2.3. **Actors and Capabilities.** As will be seen in Sect. 3.1.1, conditions are formalised as formulas in a simple language featuring variables. An actor having capability (*enabCond*, *window*), when *enabCond* holds, can give to variables appearing in *window* any value satisfying *window*. The declaration of the capabilities should be exhaustive: any action an agent is able to perform upon the system is encoded in its table of capabilities. This notion has two main particularities:

(1) The capability of an actor is conditioned by the state of the system, *i.e.* by its enabling condition. For instance, a *buyer* can engage a transaction with a *seller* provided the price for the concerned good is in a certain range of values.
(2) Then it is possible to model an actor that does not control a variable in its full range. The actor can only give this variable a certain set of values bounded by the window. It is then possible to distinguish the performances of different actors. For instance, a seller controls the delay he can ensure for the delivery to a potential client, but only within a certain window. In our example, the second hand seller can ensure the delivery in every

term between 11 and 25 days, whereas the bookshop can ensure it in every term between 4 and 25 days.

For instance, let us consider operation *commitToDeliverGood* and actor *bookshop*. The conditions of the operation are given by Table 1 and the capabilities of *bookshop* are in Table 2. In these tables and in the following we use the following abbreviations to designate the variables: $s.o$, $b.o$ and cD stand respectively for *seller.offer*, *buyer.offer* and *committedDelay*. The window $cD \in [4, 25]$ enables the bookshop to satisfy the post condition for *impatientDeliveryDelay*: $cD < 7$. But if *window* was interpreted as a classical post-condition, its validity would not ensure the satisfaction of $cD < 7$. Indeed, the post-condition would still be satisfied if, say, $cD = 12$.

Notice that the case where two actors have capabilities with overlapping enabling conditions and (at least) one common variable in their windows leads to a potential deadlock. We call such situation a case of *competing capabilities*. As we are dealing with potential choices and not effective ones, treating effective deadlocks is out of the scope of this paper.

2.4. **Assignment of roles to coalitions.** Operations appear as hexagons in Fig. 3 under leaf goals, and are gathered into roles (plain ellipses labelled by underlined circles). Notice that the methodological question of how this gathering is made by engineers is out of the scope of this paper.

Roles draw a notion of required agents, emerging from the goals specifications. And they are finally assigned to coalitions of actors.

Several coalitions can also be assigned the same role, as is the case for role *buyer* and actors *impatient* and *thrifty*. Note there are two different ways to compose actors:

(1) They may play the role together. In our example, *bookshop* and *secondHandSeller* (*sHS*) do not have capabilities to play autonomously the role *seller*. But they can play it together as a coalition. Considering the specifications of role *seller* (Table 1) on the one hand, and the capabilities of actors *bookshop* and *sHS* on the other hand, we deduce that neither actor is able to play fully role *seller*. Indeed, the bookshop cannot put its offer under 15 € and then cannot satisfy the *reqPost* condition for *thriftyInterestingPrice*: $s.o < 12$. And sHS cannot commit to deliver the good under a delay of 11 days and thus cannot satisfy the impatient condition for being delivered within 7 days. But they gather together all the required capabilities for playing the role *buyer*, which is presented with further details in Sect.17.1.
(2) And each of them may play the whole role, just as *impatient* and *thrifty* independently are able to play the role *buyer* on their own. In this case, both *thrifty* and *impatient* are unary coalitions and, as such, are separately assigned the role.

Obviously, a major criterion for the correctness of a model is what we call the assignment problem, that is the question of whether coalitions can play the roles they are assigned.

The verification procedure for the assignment problem enables the requirement engineer to identify the potential lack of capable actors. A role (or parts of roles) that is not assignable to any pre-existing actor identifies one or more fresh actors that should be introduced in the system to satisfy all goals.

3. Semantics

In this section, we give the semantics of language Khi in a logic we call ATL_{Khi}. To do so, we need the presentation of the following formal background.

3.1. **Formal background: temporal and multi-agent logics.** ATL_{Khi} is built by integrating temporal and choice operators with propositional logic. It is a fragment of the better known ATL* [2] and thus inherits its decidability.

- A language Cond_{KHI}, expressing boolean combinations of variable constraints for the description of conditions: *dom* and *req* conditions for operations as well as *enabCond* and *window* conditions for the actors' capabilities.
- The Linear Temporal Logic (LTL_{KHI}) with atoms in Cond_{KHI} for goal expression and operationalisation.
- A multi-agent logic for the mention of actors and the relation of assignment: Alternating-time Temporal Logic for KHI (ATL_{KHI}). Notably, it will enable us to express the property of a coalition of actors to support a given role.

3.1.1. *The expression of conditions, $Cond_{\text{KHI}}$.* Every condition in KHI is described in the language Cond_{KHI}, a propositional logic which atoms are comparisons of values between variables and natural numbers.

Definition 4. *Given a set of variables U, the language of $Cond_{\text{KHI}}$ over U is given by the following syntax:*

$$\varphi ::= x \sim n \mid x - y \sim n \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi$$

where $x, y \in U$, $n \in \mathbb{N}$, and $\sim \in \{<, >, =, \leq, \geq\}$.

The definition of a *window* uses a fragment of Cond_{KHI}, denoted by $\text{Cond}^{win}_{\text{KHI}}$, in which variables are explicitly bounded.

Definition 5 ($\text{Cond}^{win}_{\text{KHI}}$)**.** *$Cond^{win}_{\text{KHI}}$ is given by the following grammar:*

$$\varphi ::= a \leq x \wedge x \leq b \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$$

where $x \in U$ and $a, b \in \mathbb{N}$.

5.0.2. *The simulation of time, LTL [15].* LTL is a temporal logic in which we reason about discrete flow of time. The temporal operators used in LTL are $\circ$ and $\mathbf{U}$. Their intuitive meaning is as follows:

- $\circ\varphi$ expresses that condition φ holds in the next state from the current one.
- $\varphi_1 \mathbf{U} \varphi_2$ expresses that the condition φ_1 holds in the current state and remains true until condition φ_2 holds.

Definition 6. *The language of LTL is defined by the following syntax:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \circ\varphi \mid \varphi \mathbf{U} \varphi$$

where p ranges over a countable set P of atomic propositions. In the frame of this article the set of atomic propositions is the set of formulas in the language Cond_{KHI} and we call LTL_{KHI} that instance of LTL.

It is interpreted in (ℕ, V) where ℕ is the set of natural numbers and $V : \mathbb{N} \to 2^P$ is a function associating each n in ℕ with a subset of P. For each $n \in \mathbb{N}$, $\text{V}(n)$ is the set of atomic propositions that hold in n.

The relation of satisfaction for LTL is as follows:

Definition 7 (Semantical satisfaction for LTL)**.** *Given an interpretation function $V : \mathbb{N} \to 2^P$ and an integer $i \in \mathbb{N}$, we define the satisfaction relation by induction on the formulas:*

- *$V, i \models_{LTL} p$, for all $p \in P$, iff $p \in \text{V}(i)$.*
- *$V, i \models_{LTL} \neg\varphi$ iff $V, i \not\models \varphi$ (iff it is not the case that $V, i \models_{LTL} \varphi$).*
- *$V, i \models_{LTL} \varphi_1 \wedge \varphi_2$ iff $V, i \models_{LTL} \varphi_1$ and $V, i \models_{LTL} \varphi_2$.*
- *$V, i \models_{LTL} \circ\varphi$ iff $V, i + 1 \models_{LTL} \varphi$.*
- *$V, i \models_{LTL} \varphi_1 \mathbf{U} \varphi_2$ iff $\exists j \in V (V, i + j \models_{LTL} \varphi_2$ and $\forall k < j (V, i + k \models_{LTL} \varphi_1))$.*

We use the symbol $\Box$ as an abbreviation meaning that the formula in its scope holds at any time during the execution: $\Box\varphi := \neg(\top \mathbf{U} \neg\varphi)$

7.0.3. *Introducing agents in the system, ATL_{KHI}.* The semantics of LTL is based on a linear structure, that represents a well-determined evolution of time. In multi-agent logics we consider different possible evolutions of time. At any point in the execution, several potential evolutions are taken into account so that time has a tree-like structure. An important issue when formalising KHI is the expression of the ability of actors to ensure the satisfaction of a given LTL_{KHI} formula ψ. In other words, we need to express the ability of actors to restrict the set of potential executions so that each execution satisfies ψ. This is represented by the introduction of the operator $\langle\langle a \rangle\rangle$, where $\langle\langle a \rangle\rangle \psi$ intuitively means that the agent or coalition (*i.e.*, set) of agents a can ensure the satisfaction of ψ. This formal operation gives the language ATL_{KHI}.

ATL_{KHI} is a fragment of the larger ATL*[1]. It is decidable for the model-cheking and validity problems.

Definition 8. *The set of ATL_{KHI} formulas is given by the following grammar:*

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle a \rangle\rangle \psi$$

where p is an atomic proposition (cf Definition 6), a is a (coalition of) actor(s) and ψ is a formula in LTL_{KHI}

The operator $\langle\langle a \rangle\rangle$ acts as a complex quantifier over the set of executions: $\langle\langle a \rangle\rangle \varphi$ is true if and only if there are choices for a such that φ is true in all the executions that are compatible with these choices.

The semantics for ATL_{KHI} is given in so-called Concurrent Game Structures (CGS).

- Each state has up to countably many different successors.
- In each state, each agent has a set of available choices.
- There is a transition function: $f : S \times Ch \to S$ where S is the set of states and Ch is the set of possible choices for the different agents. It determines the transitions from each state to its successor.
- An interpretation function V associates each state with a set of atomic propositions

Definition 9. *Given a CGS M and an actor a in it, we call a strategy for a a function that maps every finite sequence of states in M ending by state s to a choice available for a in s.*

Thus a strategy for an actor gives it a choice for every situation occurring in the execution. Let us now define the satisfaction relation.

Definition 10 (Semantical satisfaction for ATL_{KHI})**.** *Let M be a CGS, s a state in it, a a coalition of actors, ψ a formula in LTL_{KHI} and φ_1, φ_2 formulas in ATL_{KHI}. Then:*

- $M, s \models_{ATL_{\text{KHI}}} p$ *iff* $p \in V(s)$
- $M, s \models_{ATL_{\text{KHI}}} \neg\varphi$ *iff it is not the case that* $M, s \models_{ATL_{\text{KHI}}} \varphi$
- $M, s \models_{ATL_{\text{KHI}}} \varphi_1 \wedge \varphi_2$ *iff* $M, s \models_{ATL_{\text{KHI}}} \varphi_1$ *and* $M, s \models_{ATL_{\text{KHI}}} \varphi_2$
- $M, s \models_{ATL_{\text{KHI}}} \langle\langle a \rangle\rangle \psi$ *iff there is a set of strategies F_a, one for each agent in a, such that any execution $\sigma = s_0, s_1, \ldots$ starting from s ($s_0 = s$) and compatible with F_a satisfies ψ ($\sigma, s_0 \models_{LTL} \psi$)*[2]*.*

10.1. **Semantics of KHI in ATL_{KHI}.** We can now give the semantics of KHI in terms of LTL_{KHI} and ATL_{KHI} formulas[3].

[1] Note that a dual operator $[\![a]\!]$ is often presented in the grammar of ATL*. It is translatable by: $[\![a]\!]\varphi$ iff $\neg\langle\langle a \rangle\rangle\neg\varphi$. We do not need it in our article and keep the notation $[\![]\!]$ for the expression of the semantics.

[2] ψ is an LTL_{KHI} formula, and is therefore interpreted on an execution, at a certain state of the execution.

[3] ATL_{KHI} is not comparable with ATL, the well-known fragment of ATL*, because the first contains formulas for semantics of role assignment of type $\langle\langle a \rangle\rangle \Box\varphi$, where φ is conjunction of formulas for *req* conditions, which are not expressible into ATL. On the other hand, ATL allows chains of nested choice operators, which are not expressible in ATL_{KHI}.

First, a goal g is expressed by a formula in LTL_{KHI}: $[\![g]\!] \in \text{LTL}_{\text{KHI}}$.

Definition 11 (Semantics of refines)**.** *The refines relation is such that the satisfaction of the set of refining goals ensures the satisfaction of the refined goals. Let g be a goal*[4]*, then* $\{[\![g.refines^{-1}]\!]\} \models_{LTL_{\text{KHI}}} [\![g]\!]$.

We follow KAOS for the formal semantics of operations and their specifications (recall that domain conditions describe the effects of an operation while required conditions are derived from the goals implemented by the operation).

Definition 12 (Semantics of operation)**.** *An operation op is defined by an occurrence of op.domPre immediately followed by an occurrence of op.domPost.*

$$[\![op]\!] := op.domPre \wedge \circ op.domPost$$

Definition 13 (Semantics of required conditions)**.** *Let op be an operation, then:*

$$\begin{array}{ll} [\![op.reqPre]\!] & := \Box([\![op]\!] \rightarrow op.reqPre) \\ [\![op.reqPost]\!] & := \Box([\![op]\!] \rightarrow \circ op.reqPost) \\ [\![op.reqTrig]\!] & := \Box((op.domPre \wedge op.reqTrig) \rightarrow [\![op]\!]) \end{array}$$

Definition 14 (Semantics of realises)**.** *The realises relation is such that the satisfaction, at every state, of all the specifications of the realising operations, ensures the satisfaction of the realised goal. Let g be a goal, then:*

$$\{[\![g.realises^{-1}.req]\!]\} \models_{LTL_{\text{KHI}}} [\![g]\!]$$

where $g.realises^{-1}.req$ stands for every conditions in $g.realise^{-1}.reqPre$, $g.realises^{-1}.reqPost$ or $g.realises^{-1}.reqTrig$.

We now come to the proper elements in language KHI, which concerns the characterisation of both concepts of agents and the relation of support that links them together.

First come the definitions of our concepts of agents. They differ by their status in the translation. A *role* is an abstract entity. Its semantics is defined in terms of the operations it provides.

Definition 15 (Semantics of role)**.** *The semantics of a role rl is the conjunction of all the semantics of conditions for the operations it provides.*

$$[\![rl]\!] := \Box \bigwedge_{r \in rl.provides.req} ([\![r]\!])$$

where rl.provides.req stands for every condition in rl.provides.reqPre, rl.provides.reqPost or rl.provides.reqTrig.

For instance, the specifications for operations provided by the role *seller* are derived in LTL_{KHI} in Table 3, from their description in Table 1.

Actors are given by ATL_{KHI} agents and coalitions by ATL_{KHI} coalitions.

Let us give the semantics of *isAbleTo*. It is quite different from a view in which the *enabCond* would simply enable the corresponding actor to satisfy the corresponding *window*. In our formalism, an actor holding (*enabCond*, *window*) as a capability not only can force *window* to hold if *enabCond* does, but in this condition he fully controls the value for some variables within the *window*.

[4]Henceforth in this article,

- we note R^{-1} for the converse of the relation R
- $\Gamma \models_{\text{LTL}_{\text{KHI}}} \varphi$ means that φ is a semantic consequence of Γ, *ie.* $\Gamma \models_{\text{LTL}_{\text{KHI}}} \varphi$ iff any structure satisfying the formulas in Γ also satisfies φ.

commitToDeliverGood	$((s.o = b.o) \to \circ(cD < 7))$
publishAd	$\wedge(\top \to \circ(s.o < 12))$

TABLE 3. Semantics of role *seller*

Definition 16 (Semantics of isAbleTo)**.** *Let a be an actor. We give the semantics of one of its capabilities $c \in a.isableTo$, and then of the whole set $a.isableTo$. At any state where $c.enabCond$ holds, a is able to give to the variables in $c.window$ any value satisfying it. Formally, let $(x_1, \dots x_k)$ be the variables in $c.window$. We call $\overline{c}$ the set of vectors $(a_1, \dots a_k) \in \mathbb{N}^k$ such that $x_1 = a_1, \dots, x_k = a_k \models_{ATL_{\text{KHI}}} c.window$. Then*

$$\begin{aligned} [\![c]\!] &:= \bigwedge_{(a_1,\dots a_k)\in\overline{c}} (\langle\!\langle a \rangle\!\rangle(\Box(c.enabCond \to \circ(x_1 = a_1 \wedge \dots \wedge x_k = a_k)))) \\ [\![a.isableTo]\!] &:= \bigwedge_{c\in a.isableTo} [\![c]\!] \end{aligned}$$

Let us now give the semantics of the relation *assignedTo*.

Definition 17 (Semantics of assignedTo)**.** *Let r be a role and $c \in r.assignedTo$ a coalition r is assigned to. Then, given the capabilities of coalition c, c is able to play role r,* i.e.*,*

$$\{[\![a.isAbleTo]\!] \mid a \in c.allies\} \models_{ATL_{\text{KHI}}} \langle\!\langle c.allies \rangle\!\rangle [\![r]\!]$$

17.1. **Application to our toy example.** Let us illustrate the definitions and the treatment of the assignment problem with our example. In the following we focus on the two different ways, mentioned in Sect.2.4, to compose actors in the assignment: either by gatering them into a coalition or by assigning them the same role.

- A coalition gathers actors together so that they are assigned a role in solidarity. This is represented by the circle in Fig. 3 between the actors *bookshop* and *sHS* and the role *seller*. Here we sketch the proof of the correction of role *seller* being assigned to coalition $\{bookshop, sHS\}$: *bookshop* and *sHS* can jointly play this role. It consists in providing a diverse offer of goods, with both cheap and quickly deliverable items. Let us also stress that neither *bookshop* nor *sHS* is able to play the role by himself. Role *seller* is given, as mentioned in Table 3, by the formula

$$\Box(((s.o = b.o) \to \circ(cD < 7)) \wedge (\top \to (\circ(s.o < 12))))$$

Table 2 shows that the bookshop is not able to propose a price under 15 €: $[\![bookshop.isAbleTo]\!] \not\models \langle\!\langle bookshop \rangle\!\rangle \Box(\circ(s.o < 12))$. Thus:

$$[\![bookshop.isAbleTo]\!] \not\models_{\text{ATL}_{\text{KHI}}} \langle\!\langle bookshop \rangle\!\rangle [\![seller]\!]$$

In a similar way, sHS cannot commit to deliver its good under 12 days so

$$[\![sHS.isAbleTo]\!] \not\models_{\text{ATL}_{\text{KHI}}} \langle\!\langle sHS \rangle\!\rangle [\![seller]\!]$$

So neither *bookshop* nor *sHS* is able to play role *seller*: they both fail on supporting one of the conditions. But since each of them is able to support the condition the other fails on, they together can ensure the satisfaction of the whole role. Indeed, *bookshop* can ensure the condition for the delay and *sHS* the condition for the price, *i.e.* we have the following situation:

$$\begin{aligned} &[\![bookshop.isAbleTo]\!] \models_{\text{ATL}_{\text{KHI}}} \langle\!\langle bookshop \rangle\!\rangle \Box((s.o = b.o) \to \circ(cD < 7)) \\ &[\![sHS.isAbleTo]\!] \models_{\text{ATL}_{\text{KHI}}} \langle\!\langle sHS \rangle\!\rangle \Box(\top \to \circ(s.o < 12)) \end{aligned}$$

Then we have:

$$[\![\{bookshop,\ sHS\}.isAbleTo]\!] \models_{\mathrm{ATL}_{\mathrm{KHI}}} \langle\!\langle bookshop\rangle\!\rangle\Box(((s.o = b.o) \to \circ(cD < 7)) \wedge (\langle\!\langle sHS\rangle\!\rangle\Box(\circ(s.o < 12))))$$

which entails $[\![\{bookshop,\ sHS\}.isAbleTo]\!] \models_{\mathrm{ATL}_{\mathrm{KHI}}} \langle\!\langle bookshop, sHS\rangle\!\rangle\Box[\![seller]\!]$.

- And several coalitions may be assigned the same role. This is the case in our example, where both unary coalitions *impatient* and *thrifty* are assigned role *buyer*. Here we only formulate the assignment problem relative to role *buyer*. To solve it one must prove that $[\![impatient.isAbleTo]\!] \models_{\mathrm{ATL}_{\mathrm{KHI}}} \langle\!\langle impatient\rangle\!\rangle[\![buyer]\!]$ and that $[\![thrifty.isAbleTo]\!] \models_{\mathrm{ATL}_{\mathrm{KHI}}} \langle\!\langle thrifty\rangle\!\rangle[\![buyer]\!]$

18. Related work

KHI gives a proposition of language for treating both the behavioural description of goal satisfaction and the mention of social agents, taken into account with their goals, capabilities and interactions.

Both items have been previously studied, but not in a coherent, semantically-rich, formal framework. KAOS proposes a semantic picture based upon temporal traces: behavioural goals are described as LTL formulas and operations as pre- and post-conditions. The notion of intentional agents is at the core of the i^* methodology. Then, the ability of agents to play roles has recently been analysed, notably in terms of commitments they make [5–7, 14].

Our work aims at unifying these achievements into a single language and enrich the semantics for RE with a temporal multi-agent language.

Furthermore, our present work enables to enrich the semantics for RE with the expression of the assignment problem, *i.e.* the very availability the actors have to support their assigned responsibilities in the description of the system. This gives two distinct perspectives for RE.

The role support itself concerns a common problem in RE which is the attribution of specifications each actor should ensure. In KAOS, it appears as the assignment of goals to agents. Nevertheless KAOS does not give any means for discussing or appreciating this question of the actual ability of the agents to ensure their assigned responsibilities.

Once the goals are refined and gathered into roles or *agent types* in TROPOS also they are assigned to actors [3]. Some extensions of TROPOS tackle the question of checking the assignment of such roles to actors [5–7]. But this checking is formalised in propositional logic. Therefore it ignores the precedence of operations as well as the mention of agents, which is made in an informal meta-language. Precedence nevertheless appears as an essential element in the formalisation of actors' actions and interactions. Actor a_1 may, for instance, be able to realise an operation o provided that a condition c is ensured. In case the said condition c is ensurable by an other actor a_2 then a_1 and a_2 are together able to satisfy c, provided that they coordinate their actions upon the system: a_2 should ensure c before a_1 performs o. Identifying synergies between actors and interdependencies thus calls for this expression of precedence.

19. Conclusion

In this paper, we have proposed a formally-rich language for RE that conciliates:

- A classical description of behavioural goals and of operations in terms of temporal logic.
- Two concepts of agents: actors as required agents and roles as prescribed agents. Actors are characterised by the goals they aim for and by capabilities and roles are characterised by operations they provide.
- A multi-agent semantics integrating both the behavioural dimension of goal satisfaction and the ability of actors to support their assigned roles in a model.

Tackling the assignment problem then gives a correctness criterion for an RE model.

Furthermore, identifying the role support can be used for giving further precisions about the specifications for the software to be introduced in a multi-agent system. KHI brings tools to identify the specified operations to be ensured and, within this set, to sort the readily ensurable ones and the ones to be provided by the software. The lasts are the very specifications of the software itself. This difference at the level of specified operations is similar to a distinction made in KAOS between leaf goals assigned to agents in the environement (*expectations*) and to agents in the software (*requirements*). Treating this distinction at the level of operations, KHI offers tools for distinguishing expected and required operations. Fig.3, eg, shows that our case study identifies an unique lacking role, called *machine* and gathering the five operations *dwellAds*, *financeAds*, *attractBuyers*, *informBuyers* and *informSellers*.

Concerning our future work, we first plan to develop fully the support for verification of KHI models. Indeed, to the best of our knowledge, there are algorithms but no available tools[5] to check the validity and perform model-checking of $\mathrm{ATL}_{\mathrm{KHI}}$ formulas. This will come with the further study of $\mathrm{ATL}_{\mathrm{KHI}}$ itself. It will in particular enable us to assess our approach concerning the assignment problem and potential deadlocks due to competing capabilities.

In this paper, we presented the semantics of KHI through a translation into $\mathrm{ATL}_{\mathrm{KHI}}$ formulas. In the future, we will directly describe the semantic model of actors and their capabilities in terms of (a fragment of) CGS. A first reason for this is to help build a more intuitive semantic picture of KHI concepts. More technically, some of the verification problems associated to an instance of KHI will then reduce to a model-checking problem (instead of the current semantic consequence problem).

Now, since the assignment problem has been successfully stated (thanks to the formal verification proposed in this paper), it is known whether all roles *can* be played by some available actors. But let us stress that they still might make other choices invalidating their assigned roles.

A natural question is then: how to distinguish between the ability of an actor to play a role and the fact that he will actually play it? A solution to this problem may be to reify actors' strategies [4] and so distinguish between an effective behaviour as a so reified strategy and a capability. A formalism with strategy would indeed enable to check the coherence between the behaviour of an actor and its assigned role. It would also enable the expression of coherence between two behaviours and then to express solutions for avoiding an effective deadlock in the case of a potential one.

REFERENCES

[1] Alur, R., Henzinger, T., Mang, F., Qadeer, S., Rajamani, S., Tasiran, S.: MOCHA: Modularity in model checking. In: Computer Aided Verification. pp. 521–525. Springer (1998)
[2] Alur, R., Henzinger, T., Kupferman, O.: Alternating-time temporal logic. In: J. ACM. pp. 672–713 (2002)
[3] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems pp. 203–236 (2004)
[4] Brihaye, T., Da Costa, A., Laroussinie, F., Markey, N.: ATL with strategy contexts and bounded memory. Logical Foundations of Computer Science pp. 92–106 (2009)
[5] Chopra, A., Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Modeling and reasoning about service-oriented applications via goals and commitments. In: Advanced Information Systems Engineering. pp. 113–128. Springer (2010)
[6] Chopra, A., Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Reasoning about agents and protocols via goals and commitments. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1. pp. 457–464. International Foundation for Autonomous Agents and Multiagent Systems (2010)

[5] The Mocha tool [1] offers facilities for the verification of ATL formulas, but not $\mathrm{ATL}_{\mathrm{KHI}}$ ones.

[7] Chopra, A., Singh, M.: Multiagent commitment alignment. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. pp. 937–944. International Foundation for Autonomous Agents and Multiagent Systems (2009)
[8] Du Bois, P.: The Albert II reference manual. Tech. rep., University of Namur (Belgium) (1997)
[9] Dubois, E., Du Bois, P., Petit, M.: ALBERT: an agent-oriented language for building and eliciting requirements for real-time systems. In: System Sciences, 1994. Vol. IV: Information Systems: Collaboration Technology Organizational Systems and Technology, Proceedings of the Twenty-Seventh Hawaii International Conference on. vol. 4, pp. 713–722. IEEE (2002)
[10] van Lamsweerde, A.: Requirements engineering, From System Goals to UML Models to Software Specifications. Wiley (2009)
[11] Letier, E., van Lamsweerde, A.: Agent-based tactics for goal-oriented requirements elaboration. In: Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on. pp. 83 –93 (may 2002)
[12] Letier, E., Van Lamsweerde, A.: Deriving operational software specifications from system goals. In: Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering. p. 128. ACM (2002)
[13] Letier, E.: Reasoning about Agents in Goal-Oriented Requirements Engineering. Ph.D. thesis, Universite Catholique de Louvain (Nov 05 2002)
[14] Mallya, A., Singh, M.: Incorporating commitment protocols into Tropos. Agent-Oriented Software Engineering VI pp. 69–80 (2006)
[15] Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Compouter Science. pp. 46–57 (1977)
[16] Silva, C., Castro, J., Tedesco, P., Araújo, J., Moreira, A., Mylopoulos, J.: Improving the architectural design of multi-agent systems: the tropos case. In: Proceedings of the 2006 international workshop on Software engineering for large-scale multi-agent systems. pp. 107–113. ACM (2006)
[17] Yu, E.: Agent-oriented modelling: software versus the world. Agent-Oriented Software Engineering II pp. 206–225 (2002)
[18] Yu, E.: Social modelling and i*. In: Conceptual Modelling: Foundations and Applications (2009)

E-mail address: FIRSTNAME.LASTNAME@ONERA.FR