A priori checking inconsistencies among strategic constraints for assembly plan generation.

Christophe Perrard, Eric Bonjour

A priori checking inconsistencies among strategic constraints for assembly plan generation

Christophe Perrard
UMR CNRS 6174 AS2M Department,
University of Franche-Comté
FEMTO-ST Institute,
24, rue Alain Savary,
25000 Besançon, France
e-mail: christophe.perrard@femto-st.fr

Eric Bonjour
ERPI (Equipe de Recherche sur les Processus Innovatifs),
University of Lorraine,
8 rue Bastien Lepage BP647,
54010 Nancy Cedex, France
e-mail: eric.bonjour@univ-lorraine.fr

Abstract.

This paper is related to the field of assembly plan generation. It describes a new approach to *a priori* check the consistency of an assembly strategy that is given by the assembly system designers before running an assembly plan generation algorithm. The aim of this work is to improve the assembly plan designer's efficiency by reducing the research space while proving the existence of acceptable solutions.

The assembly strategy combined with the product's model implies a set of constraints on the assembly processes. The proposed method determines whether the given assembly strategy produces possible assembly processes. In case of inconsistencies among the strategic constraints, the method will help the designer to identify the contradictory constraints. The set of constraints can be expressed by a Boolean equation.

First we present the key concepts and models related to the product, processes and added values in the field of assembly plan generation. Second we define existing strategic constraints, and propose three new ones and a classification of strategic assembly constraints. The originality of the proposed method consists in defining an elementary strategic constraint that is used to describe every other constraint. The proposed method leads to model an assembly strategy by a single Boolean equation that is used to check the inconsistencies. An industrial case study is provided to highlight and to demonstrate the interests of this approach.

Keywords: Assembly plan generation; assembly sequences; assembly strategy; strategic constraints; inconsistencies checking

1. Introduction

When designing an assembly system according to the technical specifications for a given product, the assembly planners try to determine feasible plans and a layout to assemble a product from its components. Two main stages of assembly planning have to be performed: first, the Assembly Sequence Planning (ASP) that generates feasible assembly sequences and second, the assembly process planning (or resource assignment, [Homem 90]) that defines the resources of the assembly system and optimizes their assignment to the assembly operations [Hui 09] [Hongmin 10] [Rashid 12]. A good assembly process plan can increase the efficiency and quality, and decrease the assembly cost and time [Zha 01] [Wang 09]. According to specialists [Homem 91] [Marian 03], an assembly sequence is the most important part of an assembly plan because it greatly affects the design of the whole assembly process: choice of resources, assembly line layout, assembly balancing, efficiency and cost. Generally, ASP consists of two major activities: assembly model and assembly sequence generation [Hui 07].

The aim of ASP is to determine the order of assembly operations in the assembly line. ASP is a typical combinatorial problem [Henrioud 91] [Marian 03] [Gu 08] which has been proved to be NP-hard. The number of feasible assembly sequences increases exponentially with the number of parts. It leads to a combinatory explosion of computations and produces a prohibitive amount of results. The traditional solving approach is structured as follows: representation, generation, feasibility, selection [Gottipolu 03]. Much research has been led in this direction [Bai 05] [Wang 09]. Related works generate possible assembly sequences, check assembly feasibility (for instance, for geometric feasibility [Hui 07]) and finally try to identify the final choice by the optimization of cost or time criteria.

Much work has been done since the early 1980s on the assembly sequences generation [Bourjault 84] [Homem 90; 91] [De Fazio 87]. First research developed structured approaches in which a sequence of questions must be answered to generate the feasible sequences. On the one hand, algorithms compute all bi-partitions of a given product, then, for all acceptable operation, the process was iterated for its constituents until reaching elementary components [Henrioud 89, 91]. On the other hand, starting from elementary components, another algorithm consists in arranging the assembly operations to produce the final assembly [Perrard 07].

[Zhao 09] presented a new formalized reasoning method for assembly sequence generation which reduces the solution space. For the last ten years, a great attention has been drawn to the development of complex algorithms

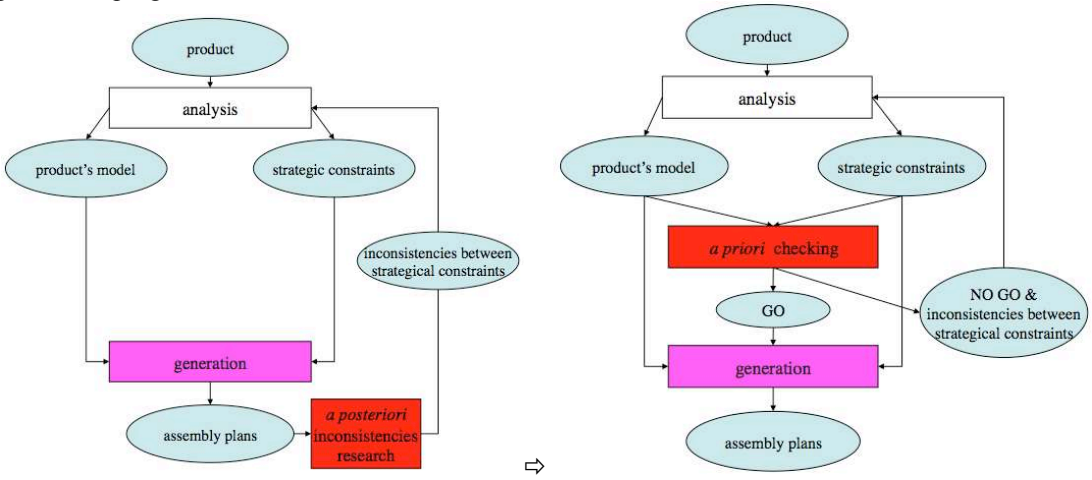
that exploit specific meta-heuristics in order to solve the ASP problem [Wang 09]. Much progress has been made to generate optimal / sub-optimal assembly sequences by developing various kinds of algorithms that have been even more sophisticated: an ant colony algorithm [Wang 05], a discrete particle swarm optimization algorithm [Lv 10] [Tseng 11], a memetic algorithm [Gao 10]. A recent review [Wang 09] highlighted that much algorithms have been based on genetic algorithms. Much research modelled the ASP problem, generated feasible assembly sequences and then selected optimal assembly sequences using genetic algorithms [Bonneville 95], [Lazzerini 00], [Lit 01], [Marian 03] or even more innovative combination with another optimisation technique [Zhou 11]. For instance, [Choi 09] optimized a multi-criteria ASP problem using genetic algorithms. However, [Lv 10] underlined that the proportion of feasible sequences in the initial population has a great effect on the performance of such algorithms. There is a need for methods that help to check the feasibility of initial sequences and then, the child sequences generated by the operations of mutation and crossover.

The previous optimisation approaches often fail to represent both product designer's and assembly planner's preferences, so that the resulting assembly sequences are not fully satisfactory from their point of view. Moreover, the presentation of all feasible assembly sequences to assembly engineers for a manual selection of good assembly sequences is not practical. A large number of unfamiliar assembly sequences simply overwhelm them [Hui 07].

Few authors have mentioned the interest of strategic constraints [Henrioud 89] [Homem 91] [Gottipolu 03] that comes from either technical or industrial concerns. These constraints are added to other design data that describe the product structure (product's model). They allow to significantly reduce the research space of assembly sequence generation algorithms and the resulting number of sequences [Martinez 09]. They are issued from an analysis that is performed by a team made of product designers and assembly planners [Demoly 11], they need assembly plans (or sequences) to define the corresponding assembly system (figure1a).

However, every inconsistency (or incompatibility) between strategic constraints leads to a long and boring procedure that may fail to produce assembly plans (figure 1a). Moreover, *a posteriori* research of inconsistencies among strategic constraints is a difficult task. Jones et al. [Jones 98] propose to deal with these inconsistencies by adding one by one each strategic constraint and by re-running the assembly plan generation algorithm each time. This is quite inefficient as regards the time spent to design new assembly plans, and when inconsistencies arise, they are not identified.

That is the reason why we propose in this paper a method that allows to check inconsistencies among strategic constraints and product's model (checking step – figure 1b). This is made before running the whole assembly plan generation program.



-a- -b-

Figure 1: main description of assembly plan generation process and proposed improvement

The proposed approach is based on the setting of elementary strategic constraints. We will prove that every other strategic constraint may be decomposed into a set of these elementary ones. Thus, only inconsistencies inside this set of elementary strategic constraints will have to be studied during the *a priori* checking step.

This paper includes four main sections:

- a brief review of the product structure model used to generate assembly plans and the review of the assembly process concept
- the review of the different strategic constraints that are usually used to deal with an assembly plan generation problem. A classification of these constraints will be proposed, in order to find out the ones

that may be used during the proposed *a priori* checking step. In this section, the strategy concept will also be defined.

- the exhaustive study of each retained strategic constraint. The elementary one will be highlighted first. This concept will be useful to validate the product structure model too.
- the description of the stages inside the step of *a priori* consistency checking. Then, a proposal will be made, in order to allow the efficient management of the strategic constraints. Some experimental results will be shown.

2. Related work

This section presents key concepts and models related to the product, processes and added values in the field of assembly plan generation. They have been mostly described in [Henrioud 89]. Notations are given in Table 1.

PF	the end product
Cp	set of the c (elementary) components to assemble. Cp={ c_1,…c_n }
VA	set of added values to provide to Cp in order to obtain the end product. VA={va_1,…va_m} ; VA= L ∪ S ∪ A
L	set of liaisons of PF. It is a sub-set of VA
S	set of attachments of PF. It is a sub-set of VA
A	set of added requirements of PF. It is a sub-set of VA
va	an element of VA
PR	the set of acceptable processes to product PF
Pr	one element of PR
SA	a sub-assembly (that exists inside at least one Pr)
Co	a constituent (that exists inside at least one Pr)
$\rceil_O O'$	the complement of sub-set O' inside set O
G1 ⊖ G2	G1-(G1∩G2) ; is read G1 « deprived of » G2
P(va_1 , va_2) or P'(VA_1 , VA_2)	strong anteriority constraint
Q(va_1 , va_2) or Q'(VA_1 , VA_2)	weak anteriority constraint
R(va_1 , va_2) or R'(VA_1 , VA_2)	anteriority constraint, either weak or strong
S(va_1 , va_2) or S'(VA_1 , VA_2)	simultaneity constraint
G({va_1…va_i…va_n})	strategic constraint of cluster ; G(C) = G({va_1…va_i…va_n}) with C ⊆ VA.
SuA({va_1…va_i…va_n})	strategic constraint of sub-assembly ; SuA(SA) = SuA({va_1…va_i…va_n}) with SA ⊆ VA.
linear(∅)	strategic constraint imposing linear processes.
linear(ℬ).	base component constraint where ℬ is the base component.

Table 1. Notations

2.1. Product model for assembly process generation

The end product may be seen, according to the assembly point of view, as the matting of two sets:

- Cp, is the set of the c (elementary) components to be assembled
- VA, is the set of va values to add to these components in order to obtain the end product. An element of VA is usually called « added value ».

The added values have the function of:

- structuration of the components together (set of liaisons L)
- perennalization of this structure (set of attachments S; an attachment is noted by the set of components and of liaisons it secures)
- and satisfaction of particular product requirements (set of auxiliary added values A, like milling, forming, checks, cleanings,…)

Example:

Let β be the product, shown by figure 2. It is made of four elementary components (A, B, C, D), four liaisons (l1, l2, l3, l4) and one screwing (V).

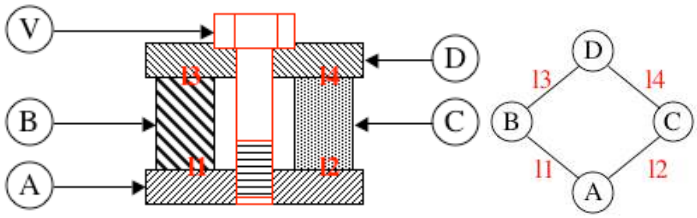


Figure 2: product β and its graph of liaisons

The corresponding model is described as follows:

Cp={A,B,C,D}

VA = L ∪ S ∪ A, with:

L={l1,l2,l3,l4} where:

l1=(A,B)

l2=(A,C)

l3=(B,D)

l4=(C,D)

S={V} where:

V=({A,B,C,D},{l1, l2, l3, l4},∅,∅)

A=Ø

Then VA = L ∪ S ∪ A = {l1 , l2 , l3 , l4 , V}

Note : in this model, the screw is considered as an attachment process, not as a component. This kind of description is more efficient than a single component description.

Typology of constituents

End product: it is the composed object that results of assembly activities (PF).

Elementary component: it is one of the initial objects that are necessary to construct the end product. It is an element of Cp.

Sub-assembly: it is an object that is produced at an intermediate step of the assembly process. A sub-assembly existence is time limited.

Constituent: it is an object that is involved in the assembly process. It can be an elementary component, a sub-assembly, or the end product PF.

Constituent model

Every constituent can be represented by a 4-tuple:

(Cp',L',S',A') where:

Cp' ⊂ Cp

L' ⊂ L

S' ⊂ S

A' ⊂ A

This describes the elements of PF (elementary components and added values) that compose this constituent. By this way:

({A,B,C,D},{l1, l2, l3, l4},{V},∅) represents the end product PF of β

({A},∅, ∅,∅) represents the elementary component A

({A,B},{l1}, ∅,∅) represents the sub-assembly that is composed of assembled components A and B

2.2. Added value, operations and process models

2.2.1. Assembly plan models

An assembly plan can be represented through different levels of description, depending on the design state of the future assembly system that is represented [Perrard 00].

a) Processes

An assembly process is the description of the partial order for assembling the product constituents. There are many ways proposed in the literature to describe an assembly process or a set of assembly processes:

- precedence graphs [Ghosh 89] [Henrioud 03] [Becker 06]
- assembly sequences (total order between added values) proposed by [Bourjault 84] and improved by [De Fazio 87] and [De Fazio 88]
- assembly trees (partial order between product constituents), proposed by [Henrioud 89, 03] and [Homem 91]

- an assembly Petri net, that simultaneously describes product constituents, operations of a set of processes proposed by [Perrard 07]

b) Functional diagrams

A functional diagram describes the functions the future assembly system will have to perform ([Perrard 93] and [Perrard 00]). Objects are constituted of product constituents and systems tools. These are oriented aggregates, where primary (fixed object) and secondary (mobile object) ones are distinguished. Operations describe elementary product transformations (added values) and logistics transformations (transfers, reorientations, components feeding,…). A functional diagram describes operations to perform, seen from the product point of view.

c) Operative diagrams

An operative diagram describes all elementary tasks the future assembly system will have to perform. An elementary task is a minimal ordered set of operations (issued from the functional diagram) that have to be performed by the same equipment. An operative diagram is a description of tasks that are considered from the equipments point of view [Lutz 94].

d) Synthesis

Most authors in the literature do not make distinction of the kind of representation they use to describe the assembly plans they propose to design. Commonly, plans are called « assembly sequences », usually without any link with the Bourjault's assembly sequence concept [Bourjault 84]. Now, in this paper, we deal with assembly processes.

2.2.2 Precedence graph of added values

We propose to represent the product added values and their associated relationship by an oriented graph. And we propose to name it « precedence graph of added values ».

The precedence graph of the added values is the model that describes each anteriority that exists between the achievements of two added values. It is an oriented graph (figure 3).

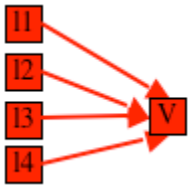
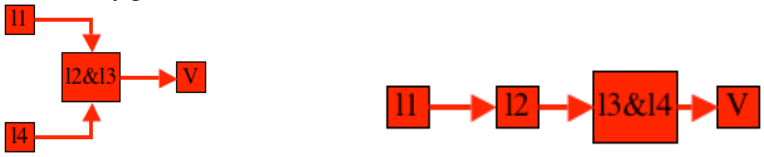


Figure 3: example of precedence graph for the product β

A precedence (or anteriority) between two added values va_1 and va_2 describes the need to perform va_1 before va_2. Obviously given precedence graph, there is a lot of processes to assemble the corresponding product.

2.2.3. Assembly process

An assembly process describes a particular way to assemble the separated elementary component in order to obtain the end product. It is made of a specific set of operations. These operations are partially ordered through a tree. A process is a particular solution that respects the precedence graph (figure 4). It is the less refined description level of an assembly plan.



-a- parallel process -b- linear process

Figure 4: examples of process for product β

Usually, a process involves:

- binary operations, if they allow to mate two constituents. They are so-called geometric operations. A geometric operation brings one (or more) liaison(s) of L. The « & » symbol indicates a multi-added values operation, like examples of figure 4.
- unary operations, if they bring an added value to an unique constituent. This added value is unique and belongs to S ∪ A. The corresponding operation, so-called non geometric, is an attachment (if the added value belongs to S) or an auxiliary one (if the added value belongs to A).
- other operations that involve more than two constituents are not taken under consideration in this paper.

Then, an assembly process contains every added value of VA and a specific partial order between them.

Thus, a process may be represented by a binary tree, where each node represents an operation. Such a tree can be represented by:

- a set of nodes O, that are the operations of the process,
- a set of oriented arcs X ⊂ O x O between the nodes of the tree. The orientation of these arcs indicates the chronological way of performing the process operations. Then, each arc is a precedence.

PR is the set of all feasible assembly processes of a given product.

2.2.4. Enrichment of operation and process concepts

An operation is partially defined when only the list of added values inside this operation is given (figure 5).

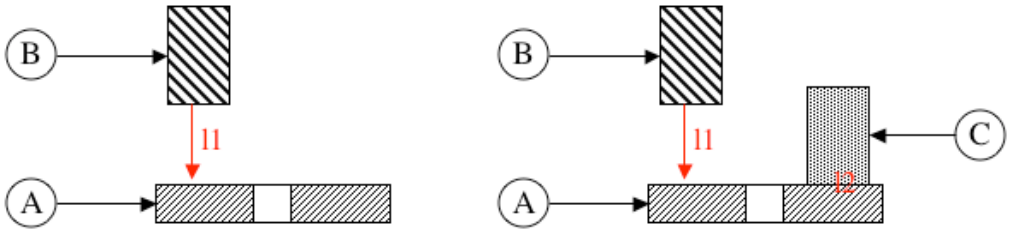
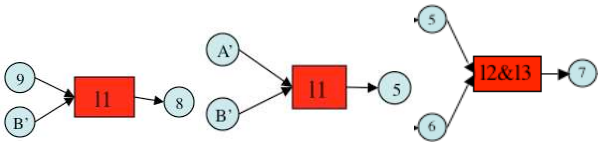


Figure 5: examples of different contexts to perform the l1 liaison of β.

It becomes fully defined if each involved sub-assembly is described: sub-assemblies before and the sub-assembly after the operation (figure 6). To completely define an operation, we must know:

- the description of each sub-assembly (C_i,L_i,S_i,A_i) that is involved by the operation (before and after it)
- the sub-set of VA brought by the operation

Thus, an operation may be represented by a 4-tuple:
((C_1,L_1,S_1,A_1), (C_2,L_2,S_2,A_2), VA_i, (C_3,L_3,S_3,A_3))



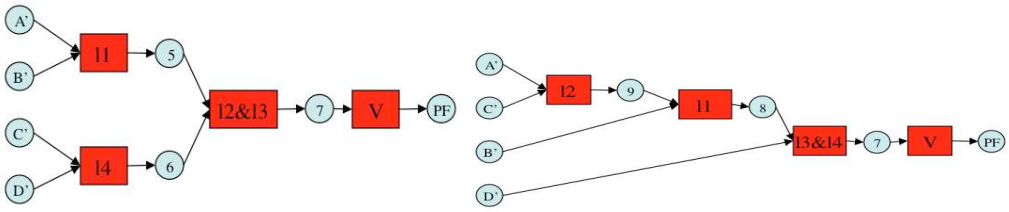
A'=({A},∅,∅,∅)
B'=({B},∅,∅,∅)
5=({A,B},{l1},∅,∅)
6=({C,D},{l4},∅,∅)
7=({A,B,C,D},{l1, l2, l3, l4},∅,∅)
8=({A,B,C},{l1, l2},∅,∅)
9=({A,C},{l2},∅,∅)

Figure 6: some examples of β product operations

Notes:

- the order between the first two terms of this 4-tuple does not matter (at this step of the assembly process design, the concept of primary/secondary sub-assemblies is not taken into account here)
- it is always possible to deduce one term of the 4-tuple from the three others. So, this 4-tuple representation introduces redundant information.

Then, any process model can be enriched with the description of each constituent involved by each operation, like figure 7:



A'=({A},∅,∅,∅)
B'=({B},∅,∅,∅)
C'=({C},∅,∅,∅)
D'=({D},∅,∅,∅)
5=({A,B},{l1},∅,∅)
6=({C,D},{l4},∅,∅)
7=({A,B,C,D},{l1, l2, l3, l4},∅,∅)
8=({A,B,C},{l1, l2},∅,∅)
9=({A,C},{l2},∅,∅)

PF=({A,B,C,D},{l1, l2, l3, l4},{V},∅)

Figure 7: examples of processes for β product including involved constituents

These two process examples (figure 7) highlight the two different contexts for the achievement of liaison l1, partially described in figure 5. The enriched processes are bi-parted trees, where:

- The first set of nodes describes the involved constituents
- The second set of nodes describes the performed operations, that allow to progress from a constituent to another.

Such a process model will be called "enriched process" in this paper.

3. Definition of strategic constraints

In this section, we define the strategic assembly constraints. Some have already been proposed and some are original. Then we propose a new classification.

3.1. Description and utility

There are two main kinds of assembly constraints:

- the operative constraints, they appear each time there is a material impossibility to perform an operation (mainly, there are the geometric constraint, the material constraint and the stability constraint) [Henrioud 89]. These constraints are revealed **during** the assembly process generation,
- the strategic constraints, that are used by the designer to describe, from his point of view and from assembly context purposes, what makes a good assembly process. These constraints are defined **before** the assembly process generation, in order to reduce the research space of the problem and to only produce "good" assembly processes.

This paper only deals with strategic constraints.

3.2. Description of already known strategic constraints

Henrioud [Henrioud 89] or Jones [Jones 98] discerned mainly three kinds of strategic constraints (the second, third and fourth below). We identified seven kinds, during our different industrial experiments.

N°1: Anteriority constraint

Such a kind of constraint is used to describe a performing order between some added values, in order to discard some difficulties to perform operations during processes generation.

N°2: Sub-assembly constraint

This kind of constraint allows the process designer to impose or to forbid some particular sub-assembly. Usually, this particular sub-assembly is required because of after-sale services needs, or to allow a particular check,… The prohibition of a sub-assembly may be done to satisfy particular safety rules, or quality requirements…

N°3: Cluster constraint

Usually, a cluster constraint is used to obtain connected added-values inside a process. These added-values imply generally common characteristics (like the use of a same tool,…) and this allows to minimize further logistical operations (minimization of the tool change number, the set-up time,…).

Connected VA: two added values (va_1 and va_2 of VA×VA) of a process are said connected either if they are already performed by two consecutive operations or simultaneously by a single operation.

the corresponding operations, that perform these added values, are connected too, or if these two are performed by the same operation.

N°4: Linear processes constraint

The designer uses this kind of constraint in order to avoid parallel building of sub-assemblies. Then, the different elementary components are added to the sub-assemblies one after the other. This is useful to obtain an assembly system architecture where the workstations are disposed along a single transfer line. This avoids the management of sub-assembly meeting during the production, notably in case of multi-products manufacturing.

3.3. Proposal of new strategic constraints

N°5: Base component constraint

This constraint complements the previous one (linear processes). It allows to choose a so called « base » component on which every other one will be added, one after the other. Usually, the base component is chosen at the beginning of the product design.

N°6: Delay or advance in the performing of an added value

This kind of strategic constraint allows the designer to include in the assembly process some particular requirements, like quality requirements, by delaying or anticipating the performing of some particular added values. For instance, the user may delay as late as possible the introduction of a fragile component assembly in order to minimize the risk of damage during the following operations (i.e. a windshield on a car). On the contrary, one may prefer to put forward a risky operation (from the quality point of view) in order to minimize the amount of added value inside the product in case of scrapping if this operation fails (i.e. a particular difficult soldering)

The reader can note that this kind of constraint is not evaluated as a Boolean criterion but through a relative comparison between the computed processes (here is computed the distance of the added value from one particular tip of the process).

N°7: Early introduction of added value

Imposing such a constraint allows to secure as best as possible the performing and the stability of produced sub-assemblies.

On the other hand, despite the whole performing conditions to be present in the considered sub-assembly, some added values (usually non geometric ones, like screwing) can be delayed inside the resulting processes, in order to group them in a further common part. This allows to minimize tool changes, for instance. However, it can generate a damage risk of the working-out product structure, especially during the performing of operations before this added-value.

However, it is seldom compatible with the cluster constraint (like performing as soon as possible an M10 screwing (this kind of constraint) and, on the other hand, delaying this screwing in order to make a cluster with other same M10 screwing and minimize tool change).

For the same reasons than the previous paragraph, this kind of constraint is not evaluated as a Boolean criterion but through a relative comparison between the resulting processes (by computing the distance in the processes between the position of actual added value and the as soon as possible position of the performing of this added value).

3.4. Classification of strategic constraints

The five first kind of strategic constraints are so-called Boolean ones, because they are either TRUE or FALSE into the candidate processes. On the other hand, the two last types of strategic constraints are so-called integer ones, because the result of their evaluation is a null or a positive integer that usually corresponds to a distance between two events of the process. The use of this result by the global strategy will make a candidate process acceptable or not (for example, conserving the only processes that produces a minimum distance,…)

3.5. Definition: assembly strategy

A product assembly strategy is:

- the set of strategic constraints that are imposed by the user to generate some particular processes
- and a global evaluation function that combines these constraints together.

Example

Let ST0 the chosen strategy for product β, where:

SC1 is a strategic constraint that expresses the user's will to perform first the bottom component of the product (for instance, by imposing the sub-assembly ({A,B,C},{l1,l2},∅,∅)) ,

SC2 is a strategic constraint that requires component A to be the base one,

SC3 is a constraint that evaluates the distance between the actual screwing performing and the presence of all of its necessary conditions

It is possible to express this strategy through a global function ST0 like:

ST0: (SC1 ∨ SC2) ∧ min(SC3)

3.6. Limits of the proposed study

The 6th and 7th constraints are more criteria than constraints, because they can be evaluated by integers (and not Boolean values). They could be used in the phase of optimised generation of assembly process. The study of this phase is out of the scope of this paper.

In this paper, we propose to limit our focus on the Boolean domain, by taking into account Boolean strategic constraints only (from the first to the fifth). Then, the global evaluation function that expresses this strategy is a Boolean equation.

4. Model of strategic constraints

In this section, the authors describe how to represent strategic constraints. We prove that each kind of strategic constraint can be expressed by a combination of anteriority. Simple examples issued from the product β design will illustrate our proposal.

4.1. Anteriority constraints

4.1.1. Introduction

Anteriority, that can exist between the performing of two added values, has not been properly defined in the literature. The concept is easy to understand when it is applied to linear operation sequences. When it is applied to trees of operations, like processes, where parallelisms and simultaneities can occur, this concept is more difficult to apprehend, as shown by table 2 (In this table, $R(va_1 , va_2)$ is the notation of an anteriority where va_1 is performed before va_2).

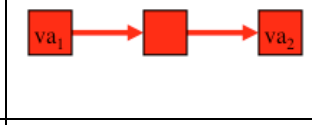
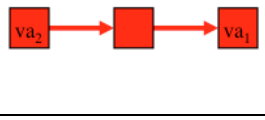
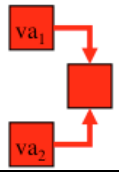
Is the anteriority verified into the process ? ↗				
Anteriority $R(va_1 , va_2)$	TRUE	FALSE	undefined	undefined

Table 2: anteriority check according to different process shapes

This leads to propose two kinds of anteriority:

- strong anteriority, that replaces the « undefined» in table 2. by "FALSE"
- and weak anteriority, that replaces the « undefined » in table 2. by "TRUE".

4.1.2. Strong anteriorities

Definition

An assembly process respects a strong anteriority constraint $P(va_1 , va_2)$ between va_1 and va_2 ($(va_1,va_2) \in VA \times VA$) if it exists an oriented path (whose distance is not null), inside this process, that starts from operation o_1 (o_1 performs va_1) and that ends at operation o_2 (o_2 performs va_2).

For a given process Pr of PR, it is possible to define the evaluation function P:

$$P: VA \times VA \rightarrow \{FALSE, TRUE\}$$
$$(va_1,va_2) \mapsto b$$

where b=TRUE if Pr respects $P(va_1 , va_2)$, else b=FALSE.

Notation

- Such a constraint is noted $P(va_1 , va_2)$.
- We propose to extend anteriority P into P' to sub-sets of VA, through this way:

$$P': \mathscr{P}(VA) \times \mathscr{P}(VA) \rightarrow \{FALSE, TRUE\}$$
$$(VA1,VA2) \mapsto b$$

b=TRUE $\Rightarrow \forall va_1 \in VA_1$ and $\forall va_2 \in VA_2$ there always is $P(va_1 , va_2)$=TRUE

($\mathscr{P}$(X) is the set of all the parties of X).

Properties

$\forall Pr \in PR$:

- $P(va_1 , va_1)$ = FALSE; the performing of an added value cannot strictly precede itself (no reflexivity). This is due to the definition above, that requires a no null distance path.
- No process can satisfy in the same time $P(va_1 , va_2)$ and $P(va_2 , va_1)$; thus $P(va_1 , va_2) \wedge P(va_2 , va_1)$ = FALSE (no symmetry)
- $P(va_1 , va_2) \wedge P(va_1 , va_2) = P(va_1 , va_2)$ (Boolean logic)
- $P(va_1 , va_2) \wedge P(va_2 , va_3) \Rightarrow P(va_1 , va_3)$ (transitivity)

Example 1: simple inconsistency detection

Let ST1: $P(l1, l2) \wedge P(l2, l3) \wedge P(l3, l1)$ the chosen strategy for the product β. Then, $\forall Pr \in PR$:

ST1$\Rightarrow$ $P(l1,l2) \wedge P(l2,l3) \wedge P(l3,l1)$

$= P(l1,l2) \wedge P(l2,l3) \wedge \underline{\mathbf{P(l1,l3)}} \wedge \underline{\mathbf{P(l3,l1)}}$ because $P(l1,l2) \wedge P(l2,l3) \Rightarrow P(l1,l3)$

$=$ FALSE

Conclusion

$\forall Pr \in PR$, ST1 $\Rightarrow$FALSE. Then, the strategy is not applicable to product β because it will provide no result.

4.1.3. Weak anteriorities

Definition

An assembly process respects a weak anteriority constraint $Q(va_1 , va_2)$ between va_1 and va_2 ($(va_1,va_2) \in VA \times VA$) if it does not exist any oriented path, whose distance is not null, inside this process, that starts from operation o_2 (o_2 performs va_2) and that ends at operation o_1.(o_1 performs va_1).

For a given process Pr of PR, it is possible to define the evaluation function Q:

$$\left| \begin{array}{l} Q: VA \times VA \rightarrow \{FALSE, TRUE\} \\ \quad (va_1,va_2) \mapsto b \end{array} \right.$$

where b=TRUE if Pr respects $Q(va_1 , va_2)$, else b=FALSE.

Notation

- Such a constraint is noted $Q(va_1 , va_2)$
- We propose to extend anteriority Q into Q' to sub-sets of VA, through this way:

$$\left| \begin{array}{l} Q': \mathscr{P}(VA) \times \mathscr{P}(VA) \rightarrow \{FALSE, TRUE\} \\ \quad (VA1,VA2) \mapsto b \end{array} \right.$$

b=TRUE $\Rightarrow \forall\ va_1 \in VA_1$ and $\forall\ va_2 \in VA_2$ there always is $Q(va_1 , va_2)$=TRUE.

Properties

As a consequence of this definition, it can be written:

$$Q(va_1, va_2) = \rceil P(va_2, va_1)$$

or more:

$\forall Pr \in PR$:

- $Q(va_1 , va_2) \vee Q(va_2 , va_1)$ = TRUE; the performing of an added value may either precede or follow the performing of another one, or they are in parallel. Every process of PR verifies one of these three cases.
- $Q(va_1 , va_2) \wedge Q(va_2 , va_1)$ = TRUE; in such a case, the two added values can only be performed either simultaneously (in the same operation) or in parallel (inside two separate branches of the process). This simultaneity $Q(va_1 , va_2) \wedge Q(va_2 , va_1)$ is noted $S(va_1 , va_2)$
- $P(va_1 , va_2) \vee P(va_2 , va_1) = \rceil S(va_1 , va_2)$

  Demonstration: $P(va_1 , va_2) \vee P(va_2 , va_1)$
  $= \rceil Q(va_2 , va_1) \vee \rceil Q(va_1 , va_2)$
  $= \rceil [\ Q(va_2 , va_1) \wedge Q(va_1 , va_2)\ ]$
  $= \rceil S(va_2 , va_1)$
- $Q(va_1 , va_1)$ = TRUE; the performing of an added value is always done in the same time as itself (reflexivity).
- $Q(va_1 , va_2) \wedge Q(va_1 , va_2) = Q(va_1 , va_2)$ (Boolean logic)
- $Q(va_1 , va_2) \wedge Q(va_2 , va_3) \Rightarrow Q(va_1 , va_3)$ (transitivity)
- $Q(va_1 , va_2) \wedge P(va_1 , va_2) \Rightarrow P(va_1 , va_2)$

Weak anteriority processing

A weak anteriority is a particular strong anteriority (complement of the inverse strong anteriority). Then, no specific processing is required. They will be included in logic computations as particular strong anteriorities.

Example 2: simultaneity implication on the process structure

Let ST2 the chosen strategy $Q(l3 , l4) \wedge Q(l4 , l3)$ for product β.

ST2$\Rightarrow$ $Q(l3,l4) \wedge Q(l4,l3)$
$= \rceil P(l4 , l3) \wedge \rceil P(l3 , l4)$
$= S(l3 , l4)$

Conclusion

The ST2 strategy is applicable to product β. In the obtained processes, the l_3 and l_4 added values are performed in a same operation during the same time, or are performed into parallel branches of the process (because none of them can precede the other). The following process (figure 8) illustrates a solution issued from the first case.

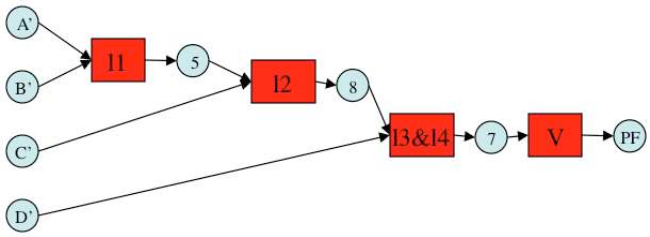


Figure 8: example of process for β product, according to ST2 strategy

Note: because of product β structure, it does not exist a process that performs the l3 and l4 added values in two distinct parallel branches. This is due to the component D, that is a common component of l_3 and l_4. Indeed, inside the acceptable processes, this imposes a single branch to perform l_3 and l_4, where D is the leave. Thus, the process shown by figure 8 is the only one that fits with ST2 strategy.

4.1.4. Notations and properties of weak and strong anteriorities

General anteriorities

The writing $R(va_1 , va_2)$ will denote a weak anteriority or a strong anteriority indifferently. It is a language abuse. For the same reason, R'(VA1 , VA2) will denote P'(VA1 , VA2) or Q'(VA1 , VA2) indifferently. This language abuse has no consequence on the validity of our proposal.

Influence domains between weak anteriority and strong anteriority

Let PR_{weak} the set of processes allowed by $Q(va_1 , va_2)$ and PR_{strong} the set of processes allowed by $P(va_1, va_2)$:
Because, $PR_{strong} \subseteq PR_{weak}$ we can write:

$$P(va_1 , va_2) \Rightarrow Q(va_1, va_2)$$
$$P(va_1 , va_2) \wedge Q(va_1, va_2) = P(va_1 , va_2)$$

Summary

Table 3. presents the application domain of strong and weak anteriorities.

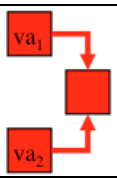
Is the anteriority verified into the process ? ↗				
Strong anteriority $P(va_1 , va_2)$	TRUE	FALSE	FALSE	FALSE
Weak anteriority $Q(va_1 , va_2)$	TRUE	FALSE	TRUE	TRUE

Table 3: strong and weak anteriority values, according to different process cases

4.1.5. Application of weak and strong anteriorities to product model checking

In a general way, each model's added value va can be expressed by a 4-tuple (Cp',L',S',A'), where:

$Cp' \subset Cp$
$L' \subset L$
$S' \subset S$
$A' \subset A$

That means, to be performed, each va' of $L' \cup S' \cup A'$ have to be performed before va. The strong anteriority concept expresses itself here. However, because Cp is not a set of added values, an element of L never implies another added value. Consequently, it is easy to write the strong anteriority, for each added value of the product model:

$$P'(L' \cup S' \cup A', \{va\})$$

Conclusion

It is now possible to link the product model compatibility to strategic constraints and then to verify them during the same processing.

Example

Let consider product β case:

$S \cup A = \{V\}$ and $V=(\{A,B,C,D\},\{l1, l2, l3, l4\},\varnothing,\varnothing)$

$\Rightarrow$ P'({l1, l2, l3, l4}, {V})

$\Rightarrow$ P(l1,V) $\wedge$ P(l2,V) $\wedge$ P(l3,V) $\wedge$ P(l4,V)

The model for all the constraints for product β is the following formula noted M:

M: P(l1,V) $\wedge$ P(l2,V) $\wedge$ P(l3,V) $\wedge$ P(l4,V)

Note

We call extended strategic formula STE, the adding of product model's constraints (M formula) to the strategic equation ST:

STE: ST $\wedge$ M

4.1.6. Process description through a set of strong anteriority constraints

The two processes for product β issued from figure 7 above can be represented by a set of strong anteriority constraints, without ambiguity, by the two following proposals:

Linear process (figure 7 - right) $\Leftrightarrow$ P(l2,l1) $\wedge$ P(l1,l3) $\wedge$ P(l1,l4) $\wedge$ P(l3,V) $\wedge$ P(l4,V)

In this representation, nothing indicates that l3 and l4 are performed simultaneously. This is not necessary because it is not possible to assemble product β through another way. However, this can be indicated by adding the following terms $\neg$P(l3,l4) $\wedge$ $\neg$P(l4,l3).

Parallel process (figure 7 - left) $\Leftrightarrow$ P(l1,l2) $\wedge$ P(l1,l3) $\wedge$ P(l4,l2) $\wedge$ P(l4,l3) $\wedge$ P(l2,V) $\wedge$ P(l3,V) $\wedge$ $\neg$P(l1,l4) $\wedge$ $\neg$P(l4,l1)

The two last terms $\neg$P(l1,l4) $\wedge$ $\neg$P(l4,l1) have to be written, in order to express that l1 and l4 must be performed in two distinct parallel branches of the process. If these two are missing, then other linear processes may satisfy the proposed equation too.

4.1.7. Conclusion

Strong anteriority constraints may be used to represent the product model and any weak anteriority. It is useful too to represent any process. They will be used in this paper as elementary strategic constraints. Now, every other strategic constraint will be decomposed into a set of strong anteriority constraints. They will be linked together by a logical equation that models the assembly strategy.

4.2. Clusters

Definition

Into an assembly process, the sub-set C of VA is a cluster if for each couple (va_1,va_2) of C × C the non oriented path of this process, that links va_1 to va_2 , is such that each operation of this path contains at least one added value va of C.

This implies that, during the assembly process, when an added value of the cluster is performed, then the entire cluster has to be achieved before starting another structure.

Notation

A strategic constraint of cluster is noted G(C) = G({va_1…va_i…va_n}) with C $\subseteq$ VA.

Example of notation: for the product β, let C1 the cluster constraint G({l1,l2})

Properties

Because of the cluster's definition, it comes:

Let C1 and C2 two clusters such C1$\cap$C2$\neq\varnothing$ then:

$\Rightarrow$ C1 $\cup$ C2 is a cluster

$\Rightarrow$ C1 $\cap$ C2 is a cluster

$\Rightarrow$ ((C1 $\ominus$ C2) $\cup$ (C1 $\cap$ C2) $\cup$ (C2 $\ominus$ C1)) is a cluster

$\Rightarrow$ ((C1 $\ominus$ C2) $\cup$ (C1 $\cap$ C2)) is a cluster

$\Rightarrow$ ((C1 $\cap$ C2) $\cup$ (C2 $\ominus$ C1)) is a cluster

However,

(C1 $\ominus$ C2) and (C2 $\ominus$ C1) are not necessarily clusters.

Model with anteriority constraints

Let the cluster constraint G({va_1,…va_i,…va_n}), where C $\subset$ VA

Let va, an added value of C and va' an added value of $\neg_{VA}$ C.

A cluster constraint imposes va' to be performed before or after the C cluster, in order to conserve the integrity of the cluster into the process. If noting R' the set of corresponding anteriorities, it comes:

R'(C , {va'}) $\vee$ R'({va'} , C)

However, this anteriority R has to be refined, in order to determine if it is a strong or weak anteriority. This can be done as follows:

For each couple (va , va') of VA×VA:

- either there is no relation between va and va' ; these added values can be performed in parallel, simultaneously or sequentially to the other ; we can write:

  $$Q'(C , \{va'\}) \vee Q'(\{va'\} , C)$$

- or there is a strong anteriority constraint between va and va', because of the product model ; these added values have to be performed one after the other, inside the processes ; and we can write:

  $$P'(C , \{va'\}) \vee P'(\{va'\} , C)$$

  according to the constraints issued from the product model (see previous paragraph). However, when the model's associated equation is missing, it is possible to use

  $$Q'(C , \{va'\}) \vee Q'(\{va'\} , C)$$

  in order to relax this second constraint to the general acceptable processes.

- or there is a simultaneity between va and va'; these added values have to be performed into the same operation, inside the processes and we can write:

  $$Q'(C , \{va'\}) \wedge Q'(\{va'\} , C) \Rightarrow S'(C , \{va'\})$$

  However, this simultaneity is not general for each computed process. It can be applied only to some particular cases. Relaxing this constraint into the first form allows it to be TRUE for each acceptable process; then:

  $$Q'(C , \{va'\}) \vee Q'(\{va'\}, C)$$

Consequently, a cluster strategic constraint will always be expressed by the same general form:

$$Q'(C , \{va'\}) \vee Q'(\{va'\}, C)$$

Thus, it is possible to describe a cluster strategic constraint through a set of strong anteriority constraints and a set of weak anteriority constraints. Then, the verification and the computing of such a kind of constraint do not require any further development.

Model example

Let the previous product β, as shown above and let impose a cluster constraint G({l1,l2}), then it is possible to express it through a set of anteriority constraints like:

$$G(\{l1,l2\}) \Rightarrow$$
$$Q'(\{l1,l2\} , \{l3\}) \vee Q'(\{l3\} , \{l1,l2\})$$
$$\wedge Q'(\{l1,l2\} , \{l4\}) \vee Q'(\{l4\} , \{l1,l2\})$$
$$\wedge Q'(\{l1,l2\} , \{V\}) \vee Q'(\{V\} , \{l1,l2\})$$

Note: the third term of this equation will not be impacted by the following developments, because the M formula (see §4.1.5) will allow its simplification.

$$G(\{l1,l2\}) \Rightarrow$$
$$(Q(l1 , l3) \wedge Q(l2 , l3)) \vee (Q(l3 , l1) \wedge Q(l3 , l2))$$
$$\wedge (Q(l1 , l4) \wedge Q(l2 , l4)) \vee (Q(l4 , l1) \wedge Q(l4 , l2))$$
$$\wedge (Q(l1 , V) \wedge Q(l2 , V)) \vee (Q(V , l1) \wedge Q(V , l2))$$
$$=$$
$$(\neg P(l3, l1) \wedge \neg P(l3,l2) \vee \neg P(l1, l3) \wedge \neg P(l2, l3))$$
$$\wedge (\neg P(l4 l1) \wedge \neg P(l4,l2) \vee \neg P(l1, l4) \wedge \neg P(l2, l4))$$
$$\wedge (\neg P(V,l1) \wedge \neg P(V,l2) \vee (\neg P(l1, V) \wedge \neg P(l2, V))$$
$$=$$
$$\{\neg P(l3, l1) \wedge \neg P(l3,l2) \wedge \neg P(l4 l1) \wedge \neg P(l4,l2) \vee \neg P(l3, l1) \wedge \neg P(l3,l2) \wedge \neg P(l1, l4) \wedge \neg P(l2,l4) \vee$$
$$\neg P(l1, l3) \wedge \neg P(l2,l3) \wedge \neg P(l4 l1) \wedge \neg P(l4,l2) \vee \neg P(l1, l3) \wedge \neg P(l2,l3) \wedge \neg P(l1, l4) \wedge \neg P(l2,l4)\}$$
$$\wedge (\neg P(V,l1) \wedge \neg P(V,l2) \vee (\neg P(l1,V) \wedge \neg P(l2, V))$$

This model presents the advantage to use only one kind of anteriority constraints. This will make the next step concerning global solving easier.

Note: as explained at the beginning of this paragraph, the set of constraints that is issued from the product model (M formula of §4.1.5) has not be taken under consideration for pedagogy reasons. Adding M gives:

$$G(\{l1,l2\}) \wedge M \Rightarrow$$
$$\{\neg P(l3,l1) \wedge \neg P(l3,l2) \wedge \neg P(l4,l1) \wedge \neg P(l4,l2) \vee \neg P(l3,l1) \wedge \neg P(l3,l2) \wedge \neg P(l1,l4) \wedge \neg P(l2,l4) \vee$$
$$\neg P(l1,l3) \wedge \neg P(l2,l3) \wedge \neg P(l4,I1) \wedge \neg P(l4,l2) \vee \neg P(l1,l3) \wedge \neg P(l2,l3) \wedge \neg P(l1,l4) \wedge \neg P(l2,l4) \}$$
$$\wedge P(l1,V) \wedge P(l2,V) \wedge P(l3,V) \wedge P(l4,V)$$

4.3 Sub-assemblies

Definition

An assembly process respects a strategic constraint of sub-assembly SA if the corresponding process contains a node that exactly represents the same content than SA.

Notation

A strategic constraint that imposes the building of a sub-assembly is noted SuA(SA) = SuA($\{va_1,\ldots va_n\}$) where SA $\subseteq$ VA.

The set of added values that are present into the sub-assembly is sufficient to define it completely. The associated elementary components are implicit.

Example of notation: for the product β case, let SA1 the imposed sub-assembly, where SA1 is ({A, B},{l1},∅,∅). Because SA1 only contains the added value L1, we note SA1={l1} and the corresponding sub-assembly constraint : SuA({l1}).

Model with anteriority constraints

Let SA the imposed sub-assembly $\{va_1,\ldots va_i,\ldots va_n\}$, with SA $\subset$ VA.

Let va, an added value of SA and va' an added value of $\rceil_{VA}$ SA.

Let $\mathscr{C}$ the set of the components of Cp that are associated to the added value va and let $\mathscr{C}'$ the set of the components of Cp that are implied by an added value va'.

For each couple (va, va') of SA×$\rceil_{VA}$ SA, we can see that:

- if $\mathscr{C} \cap \mathscr{C}' = \varnothing$, there is no relation between va and va' ; then, they may eventually be performed in parallel into the acceptable processes. In the case of a linear process, va has to be performed before va', in order to respect the sub-assembly constraint. This can be expressed by:
  Q(va , va')
- on the contrary, if $\mathscr{C} \cap \mathscr{C}' \neq \varnothing$, then it exists a temporal relation between va and va' ; they have to be performed consecutively into the acceptable processes. va has to be performed first, in order to respect the sub-assembly constraint. This is expressed by:
  P(va , va')

Then, it is possible to describe a sub-assembly strategic constraint through a set of strong anteriority constraints and a second set of weak anteriority constraints, without intersection with the first one. Thus, the verification and the computing of such a kind of constraint do not require any further development.

Model examples

Let product β, as previously described. If there is an imposed sub-assembly strategic constraint that contains SuA({l1}), then it is possible to express it through a set of anteriority constraints by the following way:

SuA({l1}) ⇒ R'({l1} , {l2,l3,l4,V})

but we need to refine the nature of each anteriority, in the following way:

Q(l1 , l4) because there is no common component between l1 and l4

P'(l1 , {l2, l3,V}) because there are common components, that are:

- the set of common components between l1 and l2 is {A},
- the set of common components between l1 and l3 is {B},
- the set of common components between l1and V is {A,B}

then it comes:

SuA({l1}) ⇒ Q(l1,l4) ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,V)
= ⌉P(l4,l1) ∧ (l1,l2) ∧ P(l1,l3) ∧ P(l1,V)

This second model has the advantage to use only one kind of anteriority constraint. That will be more useful for the global solving.

It is possible to apply the same reasoning to SuA({l4}) :

SuA({l1}) ⇒ R'({l4} , {l1,l2,l3,V}) with :

Q(l4 , l1) (no common component)

P(l4 , {l2,l3,V}) because there are common components, that are:

l4 ∩ l2 = {C}
l4 ∩ l3 = {D}
l4∩ V = {C,D}

then:

SuA({l4}) ⇒⌉P(l1,l4) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V)

4 .4. Linear processes

Definition

A process respects the strategic constraint of linearity if, inside the enriched corresponding process, each geometrical operation implies that at least one sub-assembly is composed by only one elementary component (without any added value).

Notation

A strategic constraint concerning linear processes is noted **linear(∅)**. The empty set (∅) will be explained later, when the strategic constraint fixing base component will be introduced.

Example of notation

Let ST3=**linear(∅)** the applied strategy to the product β.

Model with anteriority constraints

A linear process implies that all added values are performed into a temporal sequence without parallelism (however, simultaneities remain possible). Then, it exists an added value, depending on the process, that is performed before all the others. This added value is necessarily performed alone into the first operation of the process because the first geometrical operation is the mating of two elementary components and as such, includes a single liaison. Then, it is possible to write:

$$\text{linear}(\varnothing) \Rightarrow \forall Pr \in PR, \exists\ va_1 \in VA\ /\ \forall\ va_2 \in VA \text{ with } va_2 \neq va_1 \Rightarrow P(va_1, va_2) = TRUE$$

More understandably, inside any linear process, there is a single added value that precedes all the other ones.

Model example

The ST3 strategy imposes the following constraints:

ST3: linear(∅) ⇒
P'({l1} , {l2 , l3 , l4 ,V})
∨ P'({l2} , {l1 , l3 , l4 ,V})
∨ P'({l3} , {l1 , l2 , l4 ,V})
∨ P'({l4} , {l1 , l2 , l3 ,V})
∨ P'({V} , {l1 , l2 , l3 ,l4})

If the anteriority constraints issued from the product model (M formula §4.1.5) are added, particularly state that V follows all others, then this constraints set reduces to:

STE3: linear(∅) ∧ M ⇒
P'({l1} , {l2 , l3 , l4 ,V})
∨ P'({l2} , {l1 , l3 , l4 ,V})
∨ P'({l3} , {l1 , l2 , l4 ,V})
∨ P'({l4} , {l1 , l2 , l3 ,V})

STE3: linear(∅) ∧ M ⇒
[P(l1,l2) ∧ P(l1,l3) ∧ P(l1,l4) ∧ P(l1,V)] ∨
[P(l2,l1) ∧ P(l2,l3) ∧ P(l2,l4) ∧ P(l2,V)] ∨
[P(l3,l1) ∧ P(l3,l2) ∧ P(l3,l4) ∧ P(l3,V)] ∨
[P(l4,l1) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V)]

4.5. Base component

Definition

The base component constraint is a linear process constraint where a base component $\mathscr{B}$ is implied by the first operation of the acceptable processes.

The base component constraint is more restrictive than the linear process one.

Notation

A base component constraint where $\mathscr{B}$ is the base component, is noted **linear($\mathscr{B}$)**.

Example of notation

Let ST4: **linear(A)** the applied strategy to the product β.

Model with anteriority constraints

When a base component $\mathscr{B}$ is chosen (where $\mathscr{B} \in$ Cp), a process respects the base component strategic constraint if the constraint of linearity is imposed and if the first operation of all its enriched process involves the elementary component $\mathscr{B}$.

The added values that can be performed by the first operation of the acceptable processes form a set VA' of VA; each added value implies the base component $\mathscr{B}$. This can be expressed as follows:

$\forall Pr \in PR, \exists\ va_1 \in VA' / \mathscr{B}$ is implied by va_1 and $\forall\ va_2 \in \rceil_{VA} VA'$ with $va_2 \neq va_1 \Rightarrow P(va_1, va_2) = TRUE$

Model example

Consider the ST4 strategy. The three added values of product β that imply the component A are l1, l2 and V. This strategy implies the following constraints:

ST4: linear(A) =
P'({l1} , {l2 , l3 , l4 ,V})
∨ P'({l2} , {l1 , l3 , l4 ,V})
∨ P'({V} , {l1 , l2 , l3 ,l4})

When adding the product model constraints (M formula of §4.1.5), we can see the last term of this equation is unnecessary for the same reasons as given in the previous paragraph. Then:

STE4: linear(A) ∧ M =
{[P(l1,l2) ∧ P(l1,l3) ∧ P(l1,l4) ∧ P(l1,V)] ∨
[P(l2,l1) ∧ P(l2,l3) ∧ P(l2,l4) ∧ P(l2,V)] ∨
[P(V,l1) ∧ P(V,l2) ∧ P(V,l3) ∧ P(V,l4)]} ∧
[P(l1,V) ∧ P(l2,V) ∧ P(l3,V) ∧ P(l4,V)]

STE4: linear (A) ∧ M =
{[P(l1,l2) ∧ P(l1,l3) ∧ P(l1,l4) ∧ P(l1,V)] ∨
[P(l2,l1) ∧ P(l2,l3) ∧ P(l2,l4) ∧ P(l2,V)] } ∧
[P(l1,V) ∧ P(l2,V) ∧ P(l3,V) ∧ P(l4,V)]

4.6. Synthesis

Every Boolean strategic constraint has been modelled with strong anteriority constraints. So we have proved that an assembly strategy can be expressed by a global Boolean equation. By analysing this equation, we will perform the checking of inconsistencies that may be involved by the strategy.

5. General *a priori* checking method

In this section, we propose an original method to check the inconsistencies among Boolean strategic constraints. The global Boolean equation will be simplified and semantically analysed. The result will indicate the possibility to generate acceptable assembly plans (go/no-go decision) according to the given strategy. Three simple solving examples, issued from product β case, will be developed.

5.1. General description

Because of the unification work shown in the previous paragraphs, the product model as well as the assembly strategy can be expressed by an unique Boolean equation. Its variables represent anteriorities between the added values of the product.

Then, the proposed solving method will conduct to obtain this logical equation, to simplify it and then to analyze the validity of each term, by taking under account the meaning of the variables (figure 9).

Finally, if the result of the Boolean value is FALSE, there is at least one inconsistency in the product model and/or in the proposed assembly strategy. It will be useless to run the assembly plan generation software (Figure 1.b). It is preferable to detect where the inconsistency lies and to correct the assembly strategy.

5.2. Translation of the product model and the strategic constraints into strong anteriorities

This step has been already explained in the previous parts. Thanks to the transformation of the strategic constraints and of the product model into strong anteriorities, the result of this step is a single Boolean equation. It can be automated.

5.3. Logic simplification

This step is quite easy to process. It consists in the rewriting the initial Boolean equation under a useful form (like the first canonical form of minterms, for instance). We recommend to obtain a form like an "OR of AND" (first form of minterms), in order to simplify next steps.

This stage has to be automated, due to the complexity of the Boolean equation. This complexity mainly comes from two points:

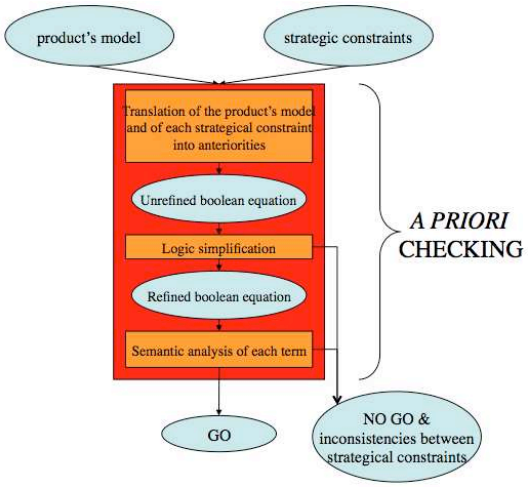


Figure 9: Main steps of an inconsistencies study

- generally, the product is quite extensive ; then, it induces a model's equation very consequent.
- some strategic constraints need, for their description, to refer to the entire set of added values concerning the product model. It leads to write very consequent equations.

Sometimes, the solving process can stop at this step, if the simplification process can determine that the result is already FALSE. This means that there is an inconsistency between strategic constraints and/or between product model and/or between the two sets of constraints. In such a case, the go/no-go variable is FALSE and the solving process stops; otherwise the solving process continues by launching the following step.

5.4. Semantic analysis

When the previous step produces a simplified Boolean equation under the form of an "OR of AND" (first form of minterms), it is necessary to analyse each term separately by using the semantic of each variable that is involved in this term.

If no term is valid according to the semantic analysis, then the general Boolean equation is not valid too. It is then possible to assume that there is at least one inconsistency.

The semantic analysis takes under consideration the meaning of the Boolean variables into each term. Fortunately, because of the unification that was previously obtained, the variables only represent strong anteriorities between product's added values.

The semantic analysis will consist, for each variable inside each term, to search a sub-set of other variables of the considered term that presents an inconsistent transitivity with the scanned one. The following examples describe such cases of inconsistency:

- $P(va1,va2) \wedge P(va2,va1)$ provides FALSE (see Section 4.1.2. Example 1)
- $P(va1,va2) \wedge P(va2,va3) \wedge P(va3,va1)$ provides FALSE

These two examples show cycles between strong anteriorities: no assembly process can have such a shape because they are trees.

- $P(va1,va2) \wedge \rceil P(va1,va2)$ provides FALSE
- $P(va1,va2) \wedge P(va2,va3) \wedge \rceil P(va1,va3)$ provides FALSE

These two examples describe a transitivity that is the complement of another variable, that is present into the same terms. Of course, it is not possible to find an assembly process that satisfies simultaneously a transitivity of strong anteriorities and its opposite.

5.5. Final result: go / no-go

The final result, called « go / no-go », represents a decision that depends on the resulting Boolean equation that comes from the previous steps. It indicates whether it is possible to generate the assembly processes (go) or not (no-go). The no-go decision means that inconsistencies have been detected in the Boolean equation.

In addition, a «no-go» result can be explained to the user, by listing and describing each inconsistency source. To do that, the implemented solution consists to label each elementary constraint (strong anteriority) with the list of identifiers of each strategic constraint that generated it. In the case of a negative solving, it provides the indication of the inconsistent terms and the set of strategic constraints that are contradictory.

5.6. Solving example 1

Let ST5 the applied strategy to the product β, that is defined as follows:

ST5 ⇒ linear(∅) ∧ SuA({l1}) ∧ SuA({l4}) = ST3 ∧ SuA({l1}) ∧ SuA({l4})

Translation of constraints and product model into strong anteriorities

The example of the ST5 strategy, that required the sub-assemblies SuA({l1}) and SuA({l4}), allowed to write the set of anteriority constraints:

SuA({l1})⇒⌉P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,V) (see Section 4.3.)

SuA({l4})⇒⌉P(l1,l4) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V) (see Section 4.3.)

and we already know:

ST3 ⇒linear(∅) ∧ M =

[P(l1,l2) ∧ P(l1,l3) ∧ P(l1,l4) ∧ P(l1,V)] ∨
[P(l2,l1) ∧ P(l2,l3) ∧ P(l2,l4) ∧ P(l2,V)] ∨
[P(l3,l1) ∧ P(l3,l2) ∧ P(l3,l4) ∧ P(l3,V)] ∨
[P(l4,l1) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V)] (see Section 4.4.)

Then, the extended strategy STE5 can be expressed by the logic equation:

ST5 ⇒

[⌉ P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,V)] ∧ [⌉ P(l1,l4) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V)] ∧
{[P(l1,l2) ∧ P(l1,l3) ∧ P(l1,l4) ∧ P(l1,V)] ∨
[P(l2,l1) ∧ P(l2,l3) ∧ P(l2,l4) ∧ P(l2,V)] ∨
[P(l3,l1) ∧ P(l3,l2) ∧ P(l3,l4) ∧ P(l3,V)] ∨
[P(l4,l1) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V)]}

Boolean equation simplification

ST5 ⇒

⌉P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3).P(l1,V) ∧ **⌉P(l1,l4)** ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V) ∧ P(l1,l2) ∧ P(l1,l3) ∧ **P(l1,l4)** ∧ P(l1,V)
∨⌉P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,V) ∧ ⌉P(l1,l4) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V) ∧ P(l2,l1) ∧ P(l2,l3) ∧ P(l2,l4) ∧ P(l2,V)
∨⌉P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,V) ∧ ⌉P(l1,l4) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V) ∧ P(l3,l1) ∧ P(l3,l2) ∧ P(l3,l4) ∧ P(l3,V)
∨**⌉P(l4,l1)** ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,V) ∧ ⌉P(l1,l4) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V) ∧ **P(l4,l1)** ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V)

(equation label 1)

The first term and the fourth term of this equation contain variables and their complement. These members are null, then the new equation is as follows:

ST5 ⇒

⌉P(l4,l1) ∧ **P(l1,l2)** ∧ P(l1,l3) ∧ P(l1,V) ∧ ⌉P(l1,l4) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V) ∧ **P(l2,l1)** ∧ P(l2,l3) ∧ P(l2,l4) ∧ P(l2,V)
∨⌉P(l4,l1) ∧ P(l1,l2) ∧ **P(l1,l3)** ∧ P(l1,V) ∧ ⌉P(l1,l4) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V) ∧ **P(l3,l1)** ∧ P(l3,l2) ∧ P(l3,l4) ∧ P(l3,V)

Semantic analysis

The two members of ST5 present Boolean terms, that are opposite (semantic analysis). Thus, the terms in the first equation's member **P(l1,l2)** and **P(l2,l1)** cancel each other out (see properties of strong anteriorities). It is the same with the second member of this equation. So, we can conclude:

ST5 produces FALSE (then, STE5 produces FALSE too).

The result of the global Boolean equation is always FALSE, no process will satisfy this strategy.

Conclusion

The STE5 strategy is not useful for the product β (The GO variable will always be FALSE).

This was predictable because the strategy requires simultaneously two distinct sub-assemblies (that leads to a parallel shape of the acceptable processes) and simultaneously, it requires a linear shape of the acceptable processes.

5.7. Solving example 2

Let ST6 the applied strategy to the product β, that is defined as follows:

ST6 ⇒ SuA({l1})∧ linear(C).

Translation of constraints and product model into strong anteriorities

The example of the previous ST5 strategy, that required the sub-assembly SuA({l1}), allowed to write the set of anteriority constraints:

SuA({l1})⇒⌉P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,V) (see Section 4.3.)

Then,

linear(C) ⇒

P'({l2} , {l1 , l3 , l4 ,V})

∨ P'({l4} , {l1 , l2 , l3 ,V})
∨ P'({V} , {l1 , l2 , l3 ,l4})
linear(C) ⇒
(P(l2,l1) ∧ P(l2,l3) ∧ P(l2,l4) ∧ P(l2,V))
∨ (P(l4,l1) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V))
∨ (P(V,l1) ∧ P(V,l2) ∧ P(V,l3) ∧ P(V,l4))

Boolean equation simplification

ST6 ∧ M ⇒
(⌉P(l4,l1) ∧ **P(l1,l2)** ∧ P(l1,l3) ∧ P(l1,V))
∧ [(**P(l2,l1)** ∧ P(l2,l3) ∧ P(l2,l4) ∧ P(l2,V))
∨ (P(l4,l1) ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V))
∨ (P(V,l1) ∧ P(V,l2) ∧ P(V,l3) ∧ P(V,l4))]
∧ P(l1,V) ∧ P(l2,V) ∧ P(l3,V) ∧ P(l4,V)
= (⌉**P(l4,l1)** ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,V))
∧ [
(**P(l4,l1)** ∧ P(l4,l2) ∧ P(l4,l3) ∧ P(l4,V))
∨ (P(V,l1) ∧ P(V,l2) ∧ P(V,l3) ∧ P(V,l4))]
∧ P(l1,V) ∧ P(l2,V) ∧ P(l3,V) ∧ P(l4,V)
= (⌉P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3) ∧ **P(l1,V))**
∧ [
(**P(V,l1)** ∧ P(V,l2) ∧ P(V,l3) ∧ P(V,l4))]
∧ P(l1,V) ∧ P(l2,V) ∧ P(l3,V) ∧ P(l4,V)
= FALSE

Semantic analysis

The result of the global Boolean equation is always FALSE. The problem is solved.

Conclusion

The ST6 strategy is not useful for the product β (The GO variable will always be FALSE).

This result was predictable because ST10 requires the component C as a base component and simultaneously, it requires the sub-assembly SuA({l1})=({A,B},{l1},∅,∅) that does not contain the component C. This can not produce a solution.

5.8. Solving example 3

Let ST7 the applied strategy to the product β.

ST7 ⇒ SuA({l1}) ∧ P(l2 , l3) ∧ P(l3 , l1)

Translation of constraints and product model into strong anteriorities

The example of the ST5 strategy (sub-assemblies), that required the sub-assembly SuA({l1}), allowed to write the set of anteriority constraints:

SuA({l1})⇒⌉P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,V) (see Section 4.3.)

Boolean equation simplification

ST7 ∧ M ⇒
⌉P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3) ∧ **P(l1,V)**
∧ P(l2,l3)
∧ P(l3,l1)
∧ **P(l1,V)** ∧ P(l2,V) ∧ P(l3,V) ∧ P(l4,V)
ST7 ∧ M ⇒
⌉P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l2,l3) ∧ P(l3,l1) ∧ P(l1,V) ∧ P(l2,V) ∧ P(l3,V) ∧ P(l4,V)

Semantic analysis

In each term of the equation, it has to be verified each precedence transitivity in order to check inconsistencies. For the actual example, there is the following inconsistency:

P(l3,l1)

that has to match with the other variables

P(l1,l2) ∧ P(l2,l3)

By transitivity, these two members provide P(l1,l3). But

P(l3,l1) ∧ P(l1,l3) = FALSE

Then, the only term of ST7 is FALSE.

Conclusion

The ST7 strategy is not useful for the product β (The GO variable will always be FALSE).

5.9. Synthesis

In this section we have explained the three steps of the proposed method for the *a priori* checking of inconsistencies. Some examples have illustrated the sequencing of these steps. When no inconsistency has been identified, the refined Boolean equation is different to FALSE. In this case, this equation admits processes that satisfy the assembly strategy.

6. Experimental results

A computer program, written in Prolog II, was created in order to validate the proposed method. The automated translation of strategic constraints, the Boolean solver and the semantic analyzer were used in order to assess the efficiency of the method.

In this section, we show the experimental results concerning the product β example and an industrial case study.

6.1. Simple example of product β

The used algorithm to generate assembly plans is the « ascending » one, as described in [Perrard 07]. The processing of the β product case generates 10 assembly plans while asking 19 questions to the user.

Consider the ST8 strategy that is defined as follows: ST8 = linear(A) ∧ SuA({l1})), we can easily prove that:

SuA({l1})⇒⎤P(l4,l1) ∧ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,V) (see Section 4.3.)

and

linear(A) ⇒
P'({l1} , {l2 , l3 , l4 ,V})
∨ P'({l2} , {l1 , l3 , l4 ,V})
∨ P'({V} , {l1 , l2 , l3 ,l4})

Then:

STE8 ⇒ P(l1,l2) ∧ P(l1,l3) ∧ P(l1,l4) ∧ P(l1,V) ∧ P(l2,V) ∧ P(l3,V) ∧ P(l4,V)

This Boolean equation is not false. Then, STE8 allows processes.

When applying the STE8 strategy, the assembly plan generation algorithm provides only two assembly plans while asking 6 questions to the user. This second experimentation enables to fully justify the efficiency of strategic constraints. This experiment shows that ST8 allows to significantly reduce the research space of assembly sequence generation algorithms and the resulting number of sequences.

When applying the ST5 strategy (ST5 = linear (∅) ∧ SuA({l1}) ∧ SuA({l4})), that was found with inconsistencies, of course, the generation algorithm provides 0 assembly plan, but asks two questions to the user. During such a run, without the use of the method proposed in this paper, the user has no indication to discover where the inconsistency comes from (from the product model ? from the strategy ? from the union of the both ?). This third trial shows the interest of the proposed method.

By analysing ST5 (see Section 5.6.) the proposed method helps to find out that the inconsistencies rely on the following incompatibilities (see equation label 1):

- 1st member: ⎤P(l1,l4) versus P(l1,l4) issued from SuA({l4}) and 1st member of linear(∅)
- 2nd member: P(l1,l2) versus P(l2,l1) issued from SuA({l1}) and 2nd member of linear(∅)
- 3rd member: P(l1,l3) versus P(l3,l1) issued from SuA({l1}) and 3rd member of linear(∅)
- 4th member: ⎤P(l4,l1) versus P(l4,l1) issued from SuA({l1}) and 4th member of linear(∅)

Then, each couple of these three strategic constraints is consistent. However, the whole set of constraints is not consistent. By tagging each anteriority with the set of constraint that generated it, the user can identify the origin of inconsistencies and correct its strategy.

6.2. Case study: electric plug

This example is issued from an industrial case study. A wall electric plug (figure 10) is composed by nine elementary components. Its assembly implies the performing of nine liaisons and four other added values (screwing, ultra-sonic soldering, crimping and final check).

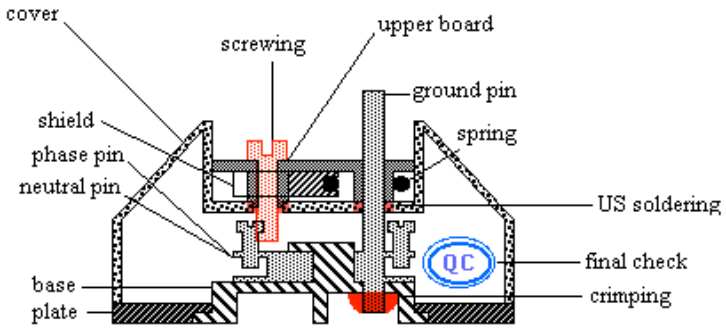


Figure 10. Electric plug

Product Model

Cp={base, plate, neutral pin, phase pin shield, cover, upper board, ground pin, spring}
VA = L ∪ S ∪ A, with:

L={l1, l2, l3, l4, l5, l6, l7, l8, l9, l10, l11}
S={Screwing, ultra-sonic soldering, crimping}
A={final check}
For readability reasons, we do not provide more details about the added values of the product's model.

Whole strategy

The proposed strategy is called « whole strategy ». It is described as follows:

a) Imposing a safety sub-assembly SA1 (figure 11) : this constraint forbids to perform the US soldering when the plate is present into the sub-assembly. It is imposed in order to ease the performing of the soldering.

b) **Imposing an electrical sub-assembly SA2 (figure 11) :** this sub-assembly belongs to different kind of plugs (round or square shaped plugs). This constraint forbids processes that do not involve the realization of the corresponding common process.

c) **Imposing an anteriority between liaisons (phase pin, base) and (ground pin, base) :** this complementary strategic constraint forbids to assemble the ground pin with the base before the phase pin. The aim is to ease the assembly of the phase pin, that is smaller than the ground pin.

d) **Imposing an anteriority between liaisons (neutral pin, base) and (phase pin, base) :** there is the same constraint between the assembly of the neutral pin and the ground pin with the base. Then, this will produce two identical processes, where the neutral pin and the phase pin are assembled before the ground pin. We propose to keep only one of them.

e) **Imposing linear processes :** this constraint only keeps processes that assemble one component after the other on a base component. This is a useful constraint to prepare an easier layout of the assembly line.

f) The respect of all of these constraints is required in the proposed strategy.

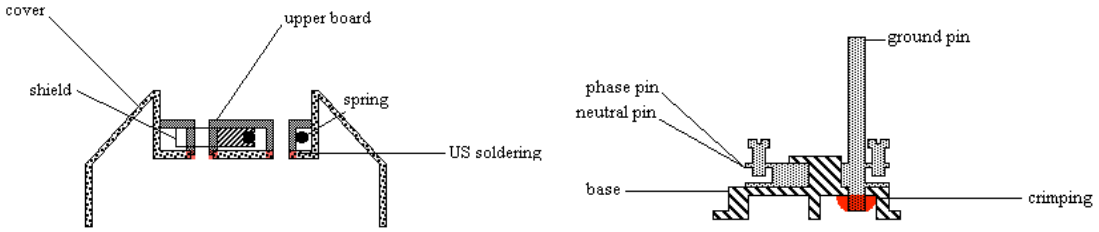


Figure 11. Safety (left) and electrical (right) sub-assemblies for the electric plug

Checking of the proposed strategy

When checking the previous strategy, the software indicates an inconsistency between a), b) and e) constraints. Then, no process will be able to satisfy the whole strategy. We do not have to run the assembly process generation software.

Explanation

Every valid process is required to respect the following three points:

- it does not exist $P(va_3, va_1)$ where $va_3 \in \rceil_{VA}SA1$ and $va_1 \in SA1$ (a) constraint)
- it does not exist $P(va_4, va_2)$ where $va_4 \in \rceil_{VA}SA2$ and $va_2 \in SA2$ (b) constraint)
- it exists va_6 where $\forall\ va_5 \in \rceil_{VA}va_5$, there is $P(va_6, va_5)$ (e) constraint)

Such a set of conditions can be satisfied only if va_5 belongs simultaneously to SA1 and SA2. However, this is not possible for the proposed strategy, because SA1 ∩ SA2 = ∅

Correction of the whole strategy

We propose to remove the last constraint e). When relaxing the whole strategy by this way (we call the result « reduced strategy »), the software does not find any other inconsistency. This allows to run the assembly process generation software.

Results (summary)

Computing assembly processes of the electric plug provides the results shown in table 4.

	Without strategy	With the whole strategy	With the reduced strategy
Number of questions	104	14	19
Number of processes	416	0	4

Table 4. Results for the electric plug

- The first case (table 4. - column 1) provides such an amount of results that it is not possible to deal with. A strategy has to be defined.
- The second case (table 4. - column 2) shows the need to check each proposed strategy. Indeed, the case of the whole strategy provides no result. However, if one runs the assembly process generation software, it will compute and ask several questions.
- The third case (table 4. - column 3) highlights the interesting results provided when applying a consistent and relevant strategy (few computations, few questions, few results that are relevant from the user's point of view).

7. Conclusion

This paper has proposed to add an *a priori* check of the consistency of assembly strategy. To do that, an exhaustive inventory of assembly constraints has been made and some new ones have been proposed. A classification of these constraints has been proposed too. Then we have focused on Boolean constraints and we have proposed an unified model by using strong anteriority constraints as elementary constraints. That results in an unique Boolean equation that can be used to check the consistency of the assembly strategy and product model. This enables to claim all Boolean inconsistencies can be discovered. Finally we have proposed to introduce this checking method to improve the efficiency of assembly plan generation.

The interest of this approach is:

- for the assembly process designer (the final user): the assembly strategy and the product's model are jointly checked. Every inconsistency is fast detected and the origin is identified. It avoids running the assembly plan generation software. This is a consequent improvement of the efficiency of assembly plan generation.
- for the product designer: he/she can give a valid assembly strategy to the assembly process designer according to the product structure.

A computer program, written in Prolog II, was created in order to validate the proposed method. The automated translation of strategic constraints, the Boolean solver and the semantic analyzer run correctly.

The next step consists in the integration of the proposed method into a module for different existing assembly plan generation softwares.

Further work will consist in unifying different *a posteriori* checking modules that already exist in many assembly plan generation software. Indeed, the proposed approach is universal due to the use of the elementary strong anteriorities. Then, the approval of any candidate operation could be based on this foundation too. Another interest concerns the design of the assembly plan generation software: a single simple module will replace a large set of specialized modules to check each applicant operation in regard to any given strategy into the generation program.

Then, it is possible to apply the consistency checking during assembly plan generation too, in order to validate each applicant operation according to the chosen strategy. This additional proposal induces huge program simplifications.

Finally, the extended strategic equation, as proposed, is a particular description of the main form of the searched assembly plans (without consideration of operative constraints). Thus, it seems to be possible to *a priori* estimate a raising number of the future computed processes. This indication is important from the user point of view, in order to evaluate the efficiency of the proposed strategy, to restrain the search space so that the running time to generate feasible assembly plans is acceptable, and without solving the whole problem.

References

[Bai 05] Bai Y.W, Chen Z.N, Bin H.Z, Hun J (2005) An effective integration approach toward assembly sequence planning and evaluation. The International Journal of Advanced Manufacturing Technology, 27(1-2):96–105

[Becker 06] Becker C, Scholl A (2006) A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research. 168(3):694-715

[Bourjault 84] Bourjault, A. (1984) Contribution à une approche méthodologique de l'assemblage automatisé: Elaboration automatique des séquences opératoires, Doctoral Thesis, University de Franche-Comte (France).

[Bonneville 95] Bonneville F, Perrard C, Henrioud J-M (1995) A genetic algorithm to generate and evaluate assembly plans. Proc ETFA '95, INRIA/IEEE Symposium on Emerging Technologies and Factory Automation, Paris, France, 10-13 Oct, 2:231-239

[Choi 09] Choi Y-K, Lee D-M, Cho Y-B (2009) An approach to multi-criteria assembly sequence planning using genetic algorithms. The International Journal of Advanced Manufacturing Technology, 42(1-2):180–188

[De Fazio 87] De Fazio T., Whitney D. (1987) Simplified Generation Of All Mechanical Assembly Sequences. IEEE Journal of Robotics and Automation. 3(6):640 – 658

[De Fazio 88] De Fazio TL, Whitney DE. (1988) Simplified generation of all mechanical assembly sequences : Corrections. IEEE Journal of Robotics and Automation RA-4(6):705-708.

[Demoly 11] Demoly F., Yan X.T.,Eynard B., Rivest L., Gomes S. (2011) An assembly oriented design Framework for product structure engineering and assembly sequence planning. Robotics and Computer-Integrated Manufacturing. Volume 27, Issue 1 :33-46

[Gao 10] Gao L, Qian W, Li X, Wang J (2010) Application of memetic algorithm in assembly sequence planning. The International Journal of Advanced Manufacturing Technology. 49(9-12):1175–1184

[Ghosh 89] Ghosh S, Gagnon R.J., (1989) A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. International Journal of Production Research. 27:637–670.

[Gottipolu 03] Gottipolu R, Ghosh K (2003) A simplified and efficient representation for evaluation and selection of assembly sequences. Computers in Industry 50:251–264

[Gu 08] Gu T, Xu Z, Yang Z (2008) Symbolic OBDD representations for mechanical assembly sequences, Computer-Aided Design 40:411–421

[Henrioud 89] Henrioud J-M (1989) Contribution à la conceptualisation de l'assemblage automatisé: nouvelle approche en vue de la détermination des processus d'assemblage. Thèse d'Etat, University of Franche-Comte (France)

[Henrioud 91] Henrioud J-M, Bourjault A (1991) LEGA: a computer-aided generator for assembly plans. Chapter 8: In Homem De Mello L-S, Lee S (1991) Computer Aided Mechanical Assembly Planning. Series: The Springer International Series in Engineering and Computer Science. Vol. 148. Hardcover, 464 p., - ISBN: 978-0-7923-9205-7

[Henrioud 03] Henrioud J-M, Relange L, Perrard C (2003) Assembly sequences, assembly constraints, precedence graphs. Proc IEEE International Symposium on Assembly and Task Planning, 10-11 July, 90-95 (ISBN: 0-7803-7770-2)

[Homem 90] Homem de Mello L.S, Sanderson A.C (1990) AND/OR graph representation of assembly plans. IEEE Transactions on Robotics and Automation 6(2):188–199.

[Homem 91] Homem De Mello L-S, Lee S (1991) Computer Aided Mechanical Assembly Planning. Series: The Springer International Series in Engineering and Computer Science. Vol. 148. Hardcover, 464 p., - ISBN: 978-0-7923-9205-7

[Hongmin 10] Hongmin Z, Dianliang W, Xiumin F (2010) Interactive assembly tool planning based on assembly semantics in virtual environment. The International Journal of Advanced Manufacturing Technology, 2010, 51(5-8):739-755

[Hui 07] Hui W, Dong X, Guanghong D, Linxuan Z (2007) Assembly planning based on semantic modeling approach. Computers in Industry 58:227–239

[Hui 09] Hui C, Yuan L, Kai-Fu Z (2009) Efficient method of assembly sequence planning based on GAAA and optimizing by assembly path feedback for complex product. The International Journal of Advanced Manufacturing Technology, 42(11-12):1187–1204

[Jones 98] Jones R.E, Wilson, R.H, Calton, T.L (1998) On constraints in assembly planning. IEEE Transactions on Robotics and Automation, 14(6):849-863

[Lazzerini 00] Lazzerini B, Marcelloni F (2000) A genetic algorithm for generating optimal assembly plans. Artificial Intelligence in Engineering, 14(4):319-329

[Lit 01] De Lit P, Latinne P, Rekiek B, Delchambre A (2001) An ordering genetic algorithm for assembly planning. International Journal of Production Research, 39(16): 3623–3640.

[Lutz 94] Lutz P, Djemel N, Bourjault A (1994) Petri net modeling for multiproducts assembly systems including testing. Proc of IEEE International Conference on Robotics and Automation, 8-13 May. San Diego, CA USA. 2 :1700-1705

[Lv 10] Lv H-G, Lu C (2010) An assembly sequence planning approach with a discrete particle swarm optimization algorithm. The International Journal of Advanced Manufacturing Technology. 50(5-8):761–770

[Marian 03] Marian R, Luong L, Abhary K (2003) Assembly sequence planning and optimisation using genetic algorithms - Part I. Automatic generation of feasible assembly sequences. Applied Soft Computing 2/3F:223–253

[Martinez 09] Martinez M, Pham V-H, Favrel J (2009) Optimal assembly plan generation: a simplifying approach. Journal of Intelligent Manufacturing 20(1):15-27

[Perrard 93] Perrard C, Djemel N, Bourjault A (1993) Design of flexible assembly systems: extension of the conceptual diagrams to the modelling of multi-product manufacturing. Proc. of International Conference on Systems, Man and Cybernetics. Le Touquet (France) 17-20 Oct. 1:526-531

[Perrard 00] Perrard C, Henrioud J.M., Vallet G (2000) Assembly systems design: a practical approch for industrial cases. 16th International Conference on CAD/CAM, Robotics and Factories of the Future, Cars & Fof 2000, June 26-28, Port of Spain, Trinidad W.I.

[Perrard 07] Perrard C, Lutz P, Salgueiro P (2007) New bottom-up algorithm for assembly plan generation: opportunities for micro-factory design. Proc IEEE International Symposium on Assembly and Manufacturing, ISAM '07, 22-25 July, 276-281

[Rashid 12] Rashid M., Hutabarat W., Tiwari A. (2012), A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. The International Journal of Advanced Manufacturing Technology, Published online 2 August 2011, Under press, DOI 10.1007/s00170-011-3499-8

[Tseng 11] Tseng Y-J., Yu F-Y., Huang F-Y. (2011) A green assembly sequence planning model with a closed-loop assembly and disassembly sequence planning using a particle swarm optimization method. The International Journal of Advanced Manufacturing Technology, 57:1183–1197

[Wang 05] Wang J.F, Liu J.H, Zhong Y.F (2005) A novel ant colony algorithm for assembly sequence planning. The International Journal of Advanced Manufacturing Technology, 25(11-12):1137–1143

[Wang 09] Wang L, Keshavarzmanesh S, Feng H-Y, Buchal R (2009) Assembly process planning and its future in collaborative manufacturing: a review. The International Journal of Advanced Manufacturing Technology 41(1-2):132–144

[Zha 01] Zha X.F, Du H.J, Qiu J.H (2001) Knowledge-based approach and system for assembly oriented design. Engineering Applications of Artificial Intelligence. 14:61–75

[Zhao 09] Zhao S, Li Z (2009) Formalized reasoning method for assembly sequences based on Polychromatic Sets theory. The International Journal of Advanced Manufacturing Technology, 42(9-10):993–1004

[Zhou 11] Zhou W., Zheng J-R., Yan J., Wang J-F. (2011) A novel hybrid algorithm for assembly sequence planning combining bacterial chemotaxis with genetic algorithm. The International Journal of Advanced Manufacturing Technology, 52:715–724