Science of Computer Programming
Manuscript Draft

Manuscript Number: SCICO-D-16-00179

Title: Strategic Reasoning for the Evaluation of the Assignment of Behavioral Goals to Agents

Article Type: SI: SBMF 2015

Keywords: requirement modelling, temporal logic, multi-agent logic.

Abstract: Given a set of behavioral goals for a system and a set of agents described with their capabilities to make the system evolve, we define the "assignment problem" as that of finding a "good" assignment of goals to (sets of) agents. To address this question, we introduce a minimal modeling framework, called Kore, as well as its semantics in terms of a strategy logic called USL. In Kore, agents are defined with their capabilities, which constrain the values of system variables, and goals are defined in terms of temporal logic formulas. Then, an assignment associates each goal with the set of agents that is responsible for its satisfaction. The problem now consists in defining and checking the correctness of this assignment. We propose different criteria to express and formalize this notion of correctness. They reduce to the satisfaction of USL formulas in an interpretation structure derived from the capabilities of agents. Thus, we end up with a procedure for deciding the correctness of the assignment. We illustrate our approach using a toy example featuring exchanges of resources between a provider and two clients.

Highlights (for review)

- The main problem that is addressed is to find a "good" assignment of goals to agents.
- A modeling framework that handles the main concepts related to this problem is proposed.
- A formal semantics is given in terms of multi-agents temporal logic.
- Checking the correctness of an assignment is then answered through model checking.

Strategic Reasoning for the Evaluation of the Assignment of Behavioral Goals to Agents[1]

Christophe Chareton[b], Julien Brunel[a,*], David Chemouil[a]

[a]*DTIM, Université fédérale de Toulouse, ONERA, F-31055 Toulouse*
[b]*LORIA, Equipe Cello, Campus Scientifique B.P. 239, 54506 Vandoeuvre-les-Nancy*

Abstract

Given a set of behavioral goals for a system and a set of agents described with their capabilities to make the system evolve, we define the "assignment problem" as that of finding a "good" assignment of goals to (sets of) agents. To address this question, we introduce a minimal modeling framework, called Kore, as well as its semantics in terms of a strategy logic called USL. In Kore, agents are defined with their capabilities, which constrain the values of system variables, and goals are defined in terms of temporal logic formulas. Then, an assignment associates each goal with the set of agents that is responsible for its satisfaction. The problem now consists in defining and checking the *correctness* of this assignment. We propose different criteria to express and formalize this notion of correctness. They reduce to the satisfaction of USL formulas in an interpretation structure derived from the capabilities of agents. Thus, we end up with a procedure for deciding the correctness of the assignment. We illustrate our approach using a toy example featuring exchanges of resources between a provider and two clients.

Keywords: Updatable Strategy Logic; Behavioral Goals; Agent Coalitions; Goal Assignment.

1. Introduction

The question of assigning behavioral goals to coalitions of agents (*i.e.* sets of active entities), the capabilities of whom are known, is a fundamental and recurring problem in various areas of software and systems engineering. We call it the *assignment problem*. In this paper, we propose a formalization of this problem and then describe various criteria to assess an assignment formally.

1.1. Motivation

First, let us illustrate various situations in which this problem arises. This will not only demonstrate the recurring character of this problem but also enable us to delineate the most salient aspects that ought to be addressed in the formalization.

In the field of Requirements Engineering (RE), several modeling languages have been proposed that each partly feature the concepts just mentioned. Thus, Kaos (Letier, 2002; Letier and Van Lamsweerde, 2002b; van Lamsweerde, 2009, 2003), a so-called *goal-oriented* modeling language, features *behavioral* goals that may be formalized using Linear Temporal Logic (LTL). This allows to assert and check the correctness of goal refinement and of goal realization (by operations). On the other hand, *agent-oriented* modeling languages, such as Tropos (Bresciani et al., 2004) and i^* (Yu, 2009, 1996), also focus on the agents that will realize these goals. A formal extension of i^*, featuring *commitments* and *protocols* (Chopra et al., 2010b,a; Chopra and Singh, 2009; Mallya and Singh, 2006), aims at checking the capabilities of agents to ensure the satisfaction of goals. In this setting, goals are described using propositional

[1]This article is a revised and extended version of (Chareton et al., 2015a).

[*]Corresponding author.
Email addresses: christophe.chareton@loria.fr (Christophe Chareton), julien.brunel@onera.fr (Julien Brunel), david.chemouil@onera.fr (David Chemouil)

logic and agents are described along with their capabilities to ensure the satisfaction of propositional formulas. Thus, the need for a treatment of the assignment problem has been identified in RE but, to the best of our knowledge, no proposition has been made until now to address it for *behavioral* goals, in a formal framework[2].

In *Systems-of-Systems Engineering* (SoSE) (Maier, 1998), several independent systems, made up of subsystems (agents in our parlance) interact altogether to achieve a global mission. Consider one of these systems and its set of agents A. Then, to investigate whether the agents in A are able to ensure a given goal in the global system, one must take into account the side effects of the actions performed by the other agents (from other systems) pursuing different goals.

Finally, in Component-Based Software Engineering (CBSE) (Szyperski, 2002), individual components (agents, for us) may be assembled into composite subsystems in order to fulfill requirements specifications. The capabilities of these agents are given as contracts (Beugnard et al., 1999). Then, knowing whether the resulting architecture indeed satisfies its specification is of major importance. Besides, identifying unsatisfied specifications can provide guidance to the engineer for adding new components. Identifying unexpected side effects (good or bad) between components is also very important.

1.2. Assignment Problem

These various examples led us to propose the following informal characterization of the assignment problem:

Definition 1 (Assignment Problem, Informally)**.** Given a set of interacting agents, the capabilities of whom are known, given a set of goals, and given an *assignment function* mapping each goal to a coalition (*i.e.* a set) of agents, is every goal assigned to a coalition that is able to ensure its satisfaction (including by leveraging actions of other coalitions)?

The objective of this paper is to refine and formalize this definition and to provide a means to solve the assignment problem.

First, we introduce Kore, a modeling framework addressing a core set of concepts needed to represent and assess the problem. Goals are described as temporal logic formulas and an agent's capability is given as a pair of a precondition and a so-called window stating how the agent can constrain the system.

It should be noted that, in Def. 1, what *interaction* is is left ambiguous. Our second contribution is precisely to propose multiple senses, each of them inducing a particular case of the problem. We call these cases *correctness criteria* for an assignment.

Finally, we formalize, using a logic called USL, the assignment problem and these criteria, and we describe a formal process to check their satisfaction for any instance.

Checking the capabilities of agents to ensure goals also enables one to distinguish, given a set of available agents and a set of goals, those goals that the available agents *cannot* ensure. The latter may then be analyzed to support the engineer in identifying *new* agents that should be introduced to fulfill all goals.

Our approach also makes it possible to characterize other sorts of interaction phenomena. Here, we stress the following:

- First, given two coalitions of agents and a goal for each coalition, we highlight *dependencies* between coalitions w.r.t. the satisfaction of their respective goals.
- Second, in an SoS for instance, we can check whether, while pursuing their own goals, agents in a system S_1 necessarily entail, as a *side effect* on the global system, that agents in a system S_2 can also ensure their own goals.

1.3. Updatable Strategy Logic

As sketched above, checking whether a correctness criterion holds will consist in assessing whether goals are assigned to coalitions of agents able to ensure them.

In practice, our approach consists in reducing such a question to a *model-checking* problem. We ask whether an interpretation structure satisfy a given formula:

[2]Note that our approach was originally developed for agent- and goal-oriented RE (Chareton et al., 2011). In this field, to the best of our knowledge, this provides the first unified formal framework addressing the satisfaction of *behavioral* goals by operation specifications and the capabilities of agents to perform these operations as required.

1. The structure is that of the possible behaviors of the system and is derived from the description of agents (and context properties).

2. The formula expresses that some coalition(s) is (are) able to ensure some goal(s) in a certain way (depending on the correctness criterion being assessed).

To achieve this, a logic that allows to reason about the ability of agents to ensure temporal properties is required. This is precisely the aim of *temporal multi-agent logics*. Here, we rely on USL (Updatable Strategy Logic), which we originally proposed to address such issues (Chareton et al., 2013, 2015b). One of the main features of USL is that agents can *compose* their behaviors (more formally: their *strategies*) in several ways, while, in other similar logics, an agent can only play following one strategy. This specificity of USL is essential here as it allows agents to play according to the different goals assigned to the coalitions they belong to.

1.4. Outline

The remainder of this article is organized as follows: in Sect. 2 we introduce the Kore modeling framework, that allows a proper representation of the situations that we wish to address. In practice, this framework can be seen as a subset of modeling languages such as Kaos or SysML. In the same section, we also introduce a minimal example which will be used in the following sections to illustrate our approach.

In Sect. 3, we give a description of the assignment problem in the framework introduced by Sect. 2. To do so, we analyze various modalities of interactions between agents and we devise corresponding correctness criteria for the assignment. This way, the assignment problem is reduced to the problem of satisfaction of the evaluation criteria by an instance of Kore.

Then, this problem is itself reduced to a model-checking problem for USL. This formalism is introduced in Sect. 4. In Sect. 5, we show how to derive an USL interpretation structure out of a Kore model.

Finally, we formalize the correctness criteria in Sect. 6 and illustrate them on our running example.

Technical Conventions and Notations Used in this Paper

- The notation for a *partial function* from a set A to a set B is $f : A \nrightarrow B$. We write $\mathrm{dom}(f)$ for the *domain of definition* of a partial function f.
- Given a binary relation $R \subseteq A \times B$, for any sets A and B, we write $R(a)$ for the set $\{b \in B \mid \langle a, b\rangle \in R\}$ and $\mathrm{dom}(R)$ for the *left projection* of R.
- Given a set S, we write $|S|$ for the *cardinality* of S, $\mathcal{P}(S)$ for the set of subsets of S, $\mathcal{P}_{>0}(S)$ for the set of non-empty subsets of S.
- Given a set S, we write S^+ for the set of non-empty finite sequences over S, and S^ω for the set of (possibly-empty) finite and infinite sequences over S.
- Given a sequence λ, we write λ_0 for its first element and λ_i for its $(i+1)$-th element. The length of a finite sequence λ (*i.e.* the number of elements in the sequence) if written $|\lambda|$ and its last element is written $\mathrm{last}(\lambda)$.
- We also write $\lambda_{\leqslant i}$ for the finite subsequence $(\lambda_0, \ldots, \lambda_i)$ of λ, and $\lambda_{\geqslant i}$ for the subsequence of λ starting at index i.

2. A Modeling Framework for the Assignment Problem

As sketched before, a Kore model specifies a set of goals to be realized, a description of the context given by various *context properties*, a set of agents (considered with their *capabilities* to act on the system) and an *assignment* of the goals to (coalitions of) agents.

As goals and context properties are temporal properties, we briefly introduce in Sect. 2.1 the logic that we use to formalize them. The Kore modeling framework is then introduced in Sect. 2.2.

2.1. LTL_{Kore}

In our setting, goals are behavioral statements: therefore we formalize them, as well as context properties, in a version of Linear Temporal Logic (LTL, (Manna and Pnueli, 1995)). This version, called LTL_{Kore}, has the following distinctive features:

- because of the multi-agent reasoning (see Sect. 4.1) we need to handle possibly finite traces, following the semantics of LTL_f (De Giacomo and Vardi, 2013) in the case of finite traces;
- the atoms of the languages are comparisons of integers terms (in Kore, the state of the system is defined by the values of integer variables);

Definition 2 (Integer terms)**.** Let X be a set of variables, the set of integer terms over X (written $\text{Iterm}(X)$) is given by the following grammar:

$$t ::= n \mid x \mid t + t \mid t - t$$

where $x \in X$, and n is a constant in $\mathbb{Z}$.

Definition 3 (LTL_{Kore} syntax)**.** Let X be a set of variables, the logic $\text{LTL}_{\text{Kore}}(X)$ is the usual Linear Temporal Logic where atoms are comparisons of integer terms. It is generated by the following grammar:

$$\varphi ::= t \sim t \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \quad \text{where } t \in \text{Iterm}(X) \text{ and } \sim \in \{<, >, =\leqslant, \geqslant\}.$$

Notation 1. The propositional connective $\vee$ ("or") and constants $\top$ ("true") and $\bot$ ("false") are defined in the obvious way. In the same way, we make use of the common temporal connector $\Box$ ("always"), defined by: $\Box\varphi \triangleq \neg(\top \,\mathsf{U}\, \neg\varphi)$.

Let us also define and introduce notations for the atomic and propositional fragments of LTL_{Kore}:

Definition 4 (*At* and Cond syntaxes)**.** Let X be a set of variables,

- The atomic fragment of LTL_{Kore} (X), written *At* (X), is the set of formulas

$$t \sim t' \quad \text{where } t, t' \in \text{Iterm}(X) \text{ and } \sim \in \{<, >, =\leqslant, \geqslant\}.$$

- The propositional fragment of LTL_{Kore} (X), written Cond (X), is generated by the following grammar:

$$\varphi ::= t \sim t \mid \neg\varphi \mid \varphi \vee \varphi \quad \text{where } t \in \text{Iterm}(X) \text{ and } \sim \in \{<, >, =\leqslant, \geqslant\}.$$

Given a set St of states, a set X of integer variables, an LTL_{Kore} formula is interpreted on a (finite or infinite) sequence λ of states, and a valuation function $val : St \times X \to \mathbb{Z}$, which associates, in each state, every variable to an integer. In the following, we extend the valuation function to any integer terms in the obvious way (noted $\overline{val}$).

Definition 5 (LTL_{Kore} satisfaction relation)**.** Given a set St of states, a set X of integer variables, a (finite or infinite) sequence λ of states, and a valuation function $val : St \times X \to \mathbb{Z}$ the satisfaction relation is defined in the following way:

- $\lambda, val \models t_1 \sim t_2$ iff $\overline{val}(\lambda, t_1) \sim \overline{val}(\lambda, t_2)$
- $\lambda, val \models \neg\varphi$ iff $\lambda, val \not\models \varphi$
- $\lambda, val \models \varphi_1 \vee \varphi_2$ iff $\lambda, val \models \varphi_1$ or $\lambda, val \models \varphi_2$
- $\lambda, val \models \mathsf{X}\varphi$ iff $|\lambda| > 1$ and $\lambda_{\geqslant i}, val \models \varphi$
- $\lambda, val \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff there is a number $i \in \mathbb{N}$ s.t. $|\lambda| > i$ and $\lambda_{\geqslant i} \models \varphi_2$ and for any $0 \leqslant j < i$, $\lambda_{\geqslant j} \models \varphi_1$

We now introduce a running example that will be used throughout this paper to illustrate our approach. In this example, we consider a resource, provided by a *prov*. Then two clients A and B have different needs w.r.t. this resource. As we will proceed through this article, we will envision three variations of this example.

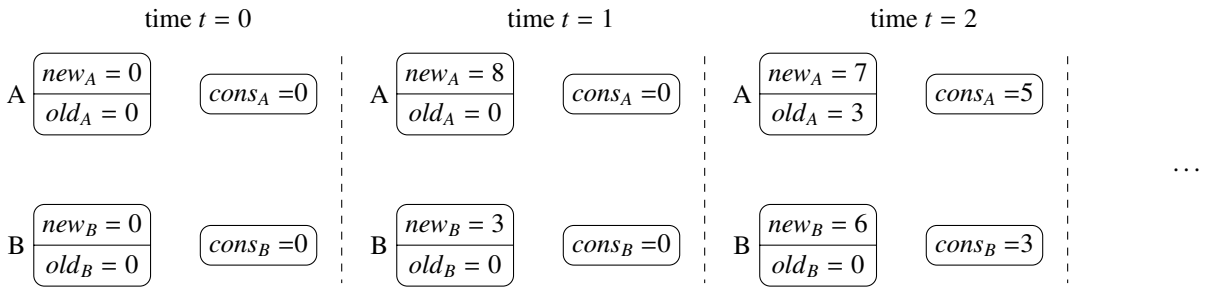


Figure 1: An example of the first states for a possible evolution of the variables in CP_1 (see Example 1).

Example 1 (CP_1)**.** In the first version (CP_1), the provider can provide up to 15 units of the resource per time unit. It can also decide to which clients the resources are affected. Concretely, at each time unit it provides the clients up to 15 new units as a whole, distributed in variables new_A for client A and new_B for client B. Each client is able to receive up to 15 units of the resource at a time.

Besides, this version of the example makes use of the following variables for every $c \in \{A, B\}$:

- at any time $t \geqslant 1$, the amount of resources available for client c is the sum of:
  - the part remaining from the resource at time $t - 1$, denoted old_c
  - the resources added by *prov* at transition from time $t - 1$ to t: new_c
- at any time $t \geqslant 2$, $cons_c$ denotes the amount of resources that has been consumed by c from the available resource ($old_c + new_c$) at time $t - 1$.

We also write X_{CP} for the set of variables $\bigcup_{c\in\{A,B\}}\{old_c, new_c, cons_c\}$.
In Fig. 1, we illustrate the use of these variables with the first transitions of a sequence example:

- at time 0, none of the resource is either produced by *prov*, available for A or B nor consumed by A or B. Therefore, the value of each variable in X_{CP} is set to 0.
- during transition from time 0 to 1, *prov* chooses to produce 11 units of the resource, 8 to A and 3 to B, received in the respective variable *new* at time 1. For each $c \in \{A, B\}$, since the store of available resources for c (*i.e.* the sum $new_c + old_c$) is empty at time 0, c cannot consume any resource during transition from time 0 to 1 and the value of $cons_c$ at time 1 is 0. Also, since this store is empty at time 0, the part remaining from time 0 at time 1 is also equal to 0.
- during transition from time 1 to 2, *prov* chooses to produce 13 units of the resource, 7 to A and 6 to B, still received in the respective variable *new*. In addition, A chooses to consume 5 units (from the ones available at time 1 in new_A and old_A) and chooses to consume 3 units. Hence the value of $cons_A$ and $cons_B$ at time 2. Finally, one can check that, for each $c \in \{A, B\}$, the value of old_c at time 2 is equal to the value of available resources for c at time 1 (which is the value of $new_c + old_c$ at time 1) minus the quantity consumed by c during transition from time 1 to 2 (*i.e.* the value of $cons_c$ at time 2). ■

2.2. *The* Kore *Modeling Framework*

Now that LTL_{Kore} has been introduced, let us delve into the concepts necessary to consider the assignment problem as we informally defined it in Sect. 1. A metamodel for Kore is given in Fig. 2.

Let us briefly introduce it. The different concepts of Kore are then detailed in the following paragraphs. First, an *agent* is characterized by various *capabilities*, each of which consists of a precondition and a window, specifying when and how this capability enables this agent to interact with the system.

Agents gather into *coalitions* in order to *realize* the *goals* that are *assigned* to them. For each instance of Kore, the *context* is defined once and for all, thanks to a number of *context properties*. The latter can be considered as

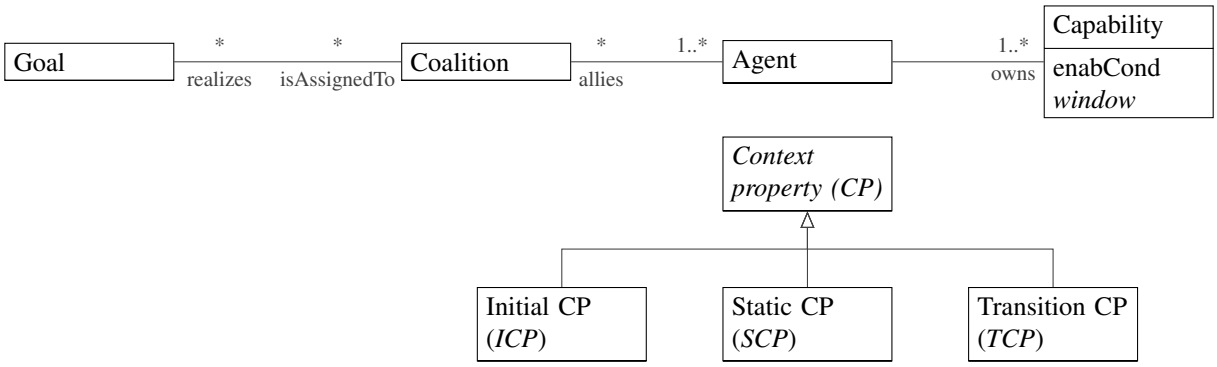


Figure 2: Metamodel of the Kore modeling framework.

global axioms of the system. They may specify the set of possible states of the system (*static context properties*), the initialization of this system (*initial context properties*) or the transitions that may occur between the different states (*transition context properties*).

Let us now give further details about each of these concepts, starting with *goals*.

2.2.1. Goals

In Kore, *goals* are statements *prescribing* the behavior expected from the system. They are formalized in LTL_{Kore}.

Example 2 (CP_1 (goals))**.** In CP_1, every client aims at consuming a certain amount of the resource: A (resp. B) wants to get at least 6 (resp. 12) at every instant $t \geqslant 2$. Note that each client has to wait for time 2 because of the provider action (which makes the resource available only from time 1) and the client action itself (getting the resource), which also takes one time unit. Their respective goals g_A and g_B are therefore formalized as follows[3]:

$$(\!|g_A|\!) \triangleq \mathsf{XX}(\Box(cons_A \geqslant 6)) \qquad (\!|g_B|\!) \triangleq \mathsf{XX}(\Box(cons_B \geqslant 12)) \qquad \blacksquare$$

2.2.2. Context Properties

Context properties are statements *describing* the system as it is (as opposed, for instance, to expected properties). In this paper:

- They may be used to specify the set of states of the system, in which case we call them *static*. A static context property *scp* is formalized as:
$$(\!|scp|\!) \triangleq \Box\varphi$$
where $\varphi \in$ Cond. (Said otherwise, a static context property specifies an invariant of the system.)
- They may also concern the initial state of the system. In this case, they are called *initial* context properties. An initial context property *icp* is formalized as:
$$(\!|icp|\!) \triangleq \varphi$$
where $\varphi \in$ Cond.
- At last, they may describe the possible transitions of the system. Then we call them *transition* context properties. A transition context property *tcp* specifies a transition from state s to s' by fixing the values of a variable x, depending of some mutually exclusive preconditions φ_k. It is formalized as:
$$(\!|tcp|\!) \triangleq \Box \bigwedge_{k\in I} (\varphi_k \rightarrow \mathsf{X}(x = t_k))$$
where I is a finite set of indices and for all $k \in I, \varphi_k \in$ Cond and $t_k \in$ Iterm.

[3] We use the notation $(\!|\cdot|\!)$ to denote the formalization of an informal property.

Example 3 (CP_1 (context))**.** In CP_1 we consider the following context properties:

Static properties $Posi_x$ All the variables in the example stand for the cardinality of some set of resources. Therefore they always have a non-negative value. For each $x \in X_{\text{CP}}$:

$$(\!|Posi_x|\!) \triangleq \Box(x \geqslant 0)$$

Lim_c The available capacity for each client is of *bounded* size PL (where PL is a constant representing the payload, say 100 for instance). For every $c \in \{A, B\}$:

$$(\!|Lim_c|\!) \triangleq \Box(new_c + old_c \leqslant \text{PL})$$

Initial properties $Init_x$ in the initial state of the system, there is none of the resource anywhere. Thus, every variable is initialized to 0. For each $x \in X_{\text{CP}}$:

$$(\!|Init_x|\!) \triangleq x = 0$$

Transition properties At any time $t \geqslant 1$, for a client c, the amount of remaining resources old_c is the amount of available available resources ($new_c + old_c$) at time $t - 1$ minus the amount consumed by c ($cons_c$) during the transition from time $t - 1$ to t :

$$(\!|Old_c|\!) \triangleq \Box(\bigwedge_{k \in 0..\text{PL}} new_c + old_c = k \rightarrow \mathsf{X}(old_c = k - cons_c))$$ ■

2.2.3. Agents and Capabilities

Agents are the active entities of the system, likely to ensure or prevent the satisfaction of goals. They are described along with their *capabilities*.

Definition 6 (Capability)**.** A capability *Cap* for an agent a is a pair of a *pre-condition* *Cap*.enabCond in Cond(X) and of a *capability window* *Cap.window*.

A capability window *window* defines a finite set of possible values for a set of variables. Formally, it is a pair of a tuple *window.dom* of n variables and a finite set *window.next* of n-tuples of values.

Intuitively, the meaning of a capability is as follows: in each state where the enabCond holds, the corresponding agent *can* constrain the *possible next states* by setting the *next value* of the variables in *window.dom* to any tuple occurring in *window.next* (a more detailed and formal explanation is given in Sect. 5). Indeed, our modeling considers that an agent able to act upon variables is not necessarily able to give them *any* value at *any* time. The enabling condition defines the conditions under which an agent can use her capability and change the values of some variables, and the window bounds the set of values she can give to these variables.

Remark 1 (Specification of window values). Along the examples developed in this article, for the sake of readability, for each window *window*, we describe *window.next* using the language Cond defined in Def. 4. The reading of this is further explained below, in Ex. 4. The careful reader will easily check that each definition of a window *window* occurring in the different versions of our case study indeed characterizes a *finite* set of tuples of possible values for the variables in *window.dom*.

Example 4 (Capabilities for the agents in CP_1)**.** The capabilities for the agents in *client-provider* are given hereafter:

Capability	Owner	enabCond	*window.dom*	*window.next*
provide	*prov*	$\top$	(new_A, new_B)	$new_A + new_B \leqslant 15$
$\{get_A^k\}_{1 \leqslant k \leqslant \text{PL}}$	A	$new_A + old_A = k$	$(cons_A)$	$cons_A \leqslant k$
$\{get_B^k\}_{1 \leqslant k \leqslant \text{PL}}$	B	$new_B + old_B = k$	$(cons_B)$	$cons_B \leqslant k$

- Capability *provide* is a capability of agent *provider*. It is effective at any state (since *provide*.enabCond = $\top$) and it enables *prov* to change the *next* value of variables new_A and new_B to any pair of values whose sum is less than or equal to 15.
- For resource consumption, A owns as many capabilities to consume the resource as the value of her payload PL. Then, if at an instant, A's storage usage if of a given size k, A can consume up to this size at the next instant thanks to capability get_B^k.
- The capabilities $\{get_B^k\}_k$ work the same way. ■

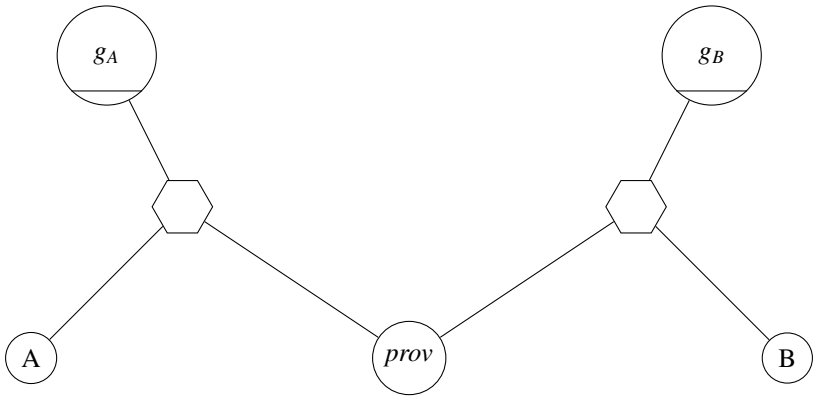


Figure 3: Assignment $\mathcal{A}_1$ for CP_1 (see Example 5). (Key: circles with bottom chords are goals, plain circles are agents, and hexagons are coalitions.)

2.2.4. Assignment

An assignment maps each goal to a coalition (*i.e.* a nonempty set) of agents. The assignment problem consists in verifying that for each goal, the coalition associated with this goal is able to ensure satisfaction of the latter, given a certain level of interaction with the other coalitions. Varying this level of interaction yields various correctness criteria.

Example 5 (CP_1 (assignment))**.** See Fig. 3, which represents the assignment $\mathcal{A}_1$ for CP_1. The assignment $\mathcal{A}_1$ assigns g_A to the coalition made up of *prov* and A, and g_A to the coalition made up of *prov* and B. ■

3. Evaluation Criteria for Assignment

Now that the different elements of Kore are defined, let us come back to the assignment problem and seek more precise criteria modeling the notion of correctness for an assignment. We first give an informal definitions for these criteria. Their formalization is given in Sect. 6, as it first requires the introduction of the USL framework (in Sect. 4) and the production of a USL interpretation structure out of a Kore model (presented in Sect. 5).

3.1. Local correctness

The first version of the assignment problem we consider is the question of whether *each goal is assigned to a coalition able to ensure it, whatever the other agents do*. We call this criterion the *local correctness* of the system under consideration.

In the *client-provider* example, this corresponds to the question of whether $\{prov, A\}$ is able to ensure the satisfaction of g_A (whatever B does) and $\{prov, B\}$ is able to ensure the satisfaction of g_B (whatever A does).

We write $LC_{\mathcal{A}}(G)$ for the satisfaction of the local correctness of a set of goals G under an assignment $\mathcal{A}$.

3.2. Global correctness

The criterion of local correctness is easy to understand and to check. Nevertheless it is not sufficient when, as is the case of CP_1, one agent is part of several coalitions being assigned different goals. Indeed, in CP_1, *prov* is able to take part separately in both coalitions $\{prov, A\}$ and $\{prov, B\}$. But the local correctness does not say whether *prov* is able to take part in both coalitions *at the same time*. What *prov* has to do with the first coalition might be contradictory with what she has to do within the second coalition.

To overcome this issue, we introduce a second correctness criterion, called the *global correctness*. Global correctness is satisfied if there is a *general behaviour* b of all agents s.t. for each goal g, knowing that the coalition of agents assigned to g behaves according to b is enough to ensure g, whatever the other agents do.

The notion of such a *general behaviour* will be defined as a *multi-strategy profile* in a CGS, in Sect. 6.

If the assignment $\mathcal{A}$ of a set of goals G is globally correct, then we write $GC_{\mathcal{A}}(G)$.

3.3. Collaboration

The global correctness criterion is sufficient to ensure that each goal is assigned to a capable coalition. Nevertheless, it may require more than what is necessary. Indeed, it requires that each coalition is able to ensure its goal in a completely autonomous way (whatever the agents not in this coalition do).

In certain cases, it may be necessary to loosen this criterion and to admit that some given coalitions depend on others to ensure their goals. To illustrate this point, let us slightly modify our example and consider a second version CP_2.

Example 6 (CP_2)**.** It brings the following changes from CP_1:

- *prov* can now produce up to 20 units at a time;
- In the new assignment $\mathcal{A}_2$, *prov* is assigned a new goal g_{prov}, which is to produce at least 16 units of the resource per time unit.
- Furthermore, g_A and g_B are respectively assigned to $\{A\}$ and $\{B\}$. Fig. 4 illustrates the goals occuring in CP_2 and the assignment $\mathcal{A}_2$. Notice that for each goal represented in Fig. 4, the goal is also labeled by its semantics.
- The context properties for CP_2 are the same as those for CP_1 (detailed in Ex. 3).
- The capabilities for agents in CP_2 are given hereafter.

Capabilities for the agents in CP_2:

Capability	Owner	enabCond	*window.dom*	*window.next*
provide	*prov*	$\top$	(new_A, new_B)	$new_A + new_B \leqslant 20$
$\{get_A^k\}_{1 \leqslant k \leqslant \text{PL}}$	A	$new_A + old_A = k$	$(cons_A)$	$cons_A \leqslant k$
$\{get_B^k\}_{1 \leqslant k \leqslant \text{PL}}$	B	$new_B + old_B = k$	$(cons_B)$	$cons_B \leqslant k$

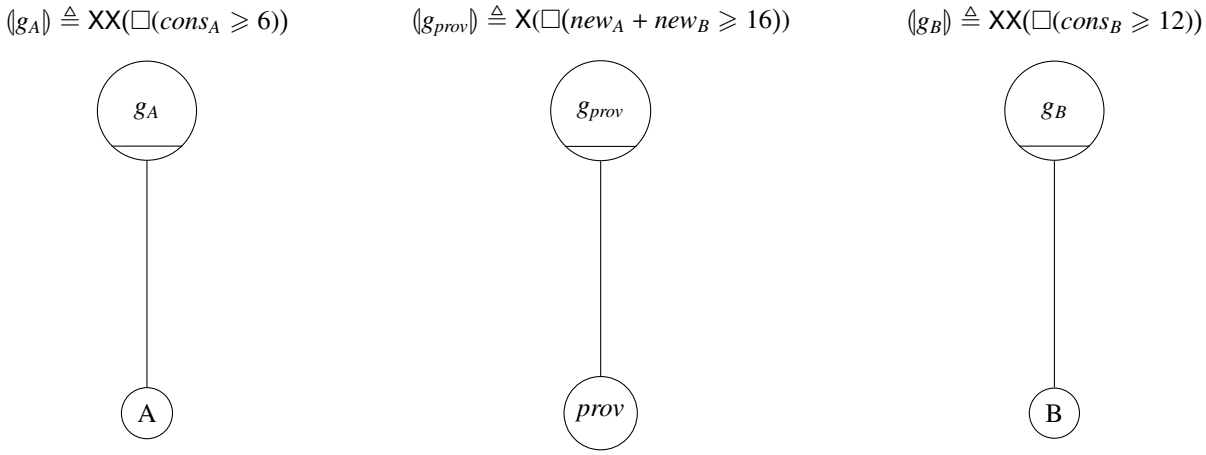


Figure 4: Goals for CP_2 and assignment $\mathcal{A}_2$.

In this second version, *prov* is at the same time able to ensure the satisfaction of her goal and to help A and B. By producing, for example, at least 7 units in new_A and 13 units in new_B at any time, she ensures g_{prov} as well as the global correctness of the model reduced to $\{g_A, g_B\}$.

In this case, we say that g_{prov} *globally collaborates* to the satisfaction of $\{g_A, g_B\}$, written $Coll_{\mathcal{A}_2}(g_{prov}, \{g_A, g_B\})$. (Note that a relation of *local collaboration* could also be defined a similar way.) ■

3.4. Contribution

In the three criteria introduced above, we adopted the point of view of an engineer controlling every agent in the system. Thus, we considered our model as a closed system and only asked whether it is possible for a unique decision-maker to specify the agents so that all goals are ensured.

In the case of open systems, the engineer of one system does not control the other systems, that interact with it. Therefore, a relevant question from the point of view of this engineer is that of whether the agents from the other interacting systems, by ensuring their goals, necessarily have favorable side effects on its model. This is what we call *contribution*. Let us consider a last version of our example, CP_3.

Example 7 (CP_3)**.** In this version, A has priority over B for the consumption of the resource: when the provider provides resources in new_A at time t, A can consume some of them and then the remainder is sent to new_B (so that, at time $t+1$, B can take some).

In order to model this, we allow agents to interact with the system only at some given states. This alternation between the agents' turns is encoded using a new variable *turn*. It is equal to 0 when *prov* or B is able to play and to 1 when it is A's turn.

We also call an a-transition (resp. b-transition) a transition from a state where $turn = 1$ to a state where $turn = 0$ (resp. a transition from a state where $turn = 0$ to a state where $turn = 1$).

Here, the provider is able to produce up to 20 units of the resource per b-transition, her goal g_{prov} is now to provide at least 18 units of the resource per b-transition and g_A and g_B are both assigned to $\{A, B\}$ in the assignment $\mathcal{A}_3$.

We illustrate this in Fig. 5, showing the first transitions of a sequence instance:

- At time 0, none of the resource is either produced by *prov*, available for A or B nor consumed by A or B. Therefore, the value of each variable in X_{CP} is set to 0.
- The transition from time 0 to 1 is a b-transition, giving turn to *prov* and B. During this transition *prov* chooses to produce 19 units of the resource, they are all delivered to A in variable new_A at time 1.

  Since B's store of available resources (*i.e.* the sum $new_B + old_B$) is empty at time 0, B cannot consume any resource during this transition and the value of $cons_B$ at time 1 is 0.

  Besides, since it is B's turn, A cannot consume any resource either.
- The transition from time 1 to 2 is an a-transition, giving turn to A. Teh agent *prov* cannot produce any of the resource. At time 1, A has 19 units available, in variable new_A. So she can consume up to 19 units.

  She chooses to consume 6 units, which appear in variable $cons_A$ at time 2. The remainder at time 1 (13 units) is made available for B at the next transition and stored in variable new_B.
- The transition from time 2 to 3 is again a b-transition, giving turn to *prov* and B. This time, *prov* chooses to produce 17 units, all delivered to A in variable new_A at time 3.

  Since it is not her turn, A cannot play during this transition and at time 3, the volume of consumed resources for her ($cons_A$) is equal to 0.

  At time 2, B chooses to consume 12 units (they appear in $cons_B$ at time 3). Then at time 3, the remaining quantity (1 unit) is stored in variable old_B.

The set X'_{CP} of variables occuring in CP_3 is defined by adding *turn* to the set X_{CP} and suppressing variable old_A since, at each a-transition, the quantity of resource that is not consumed by A is made available to B in variable new_B.

Set of variable used in CP_3

$$X'_{\text{CP}} \triangleq X_{\text{CP}} \backslash \{old_A\} \cup \{turn\}$$

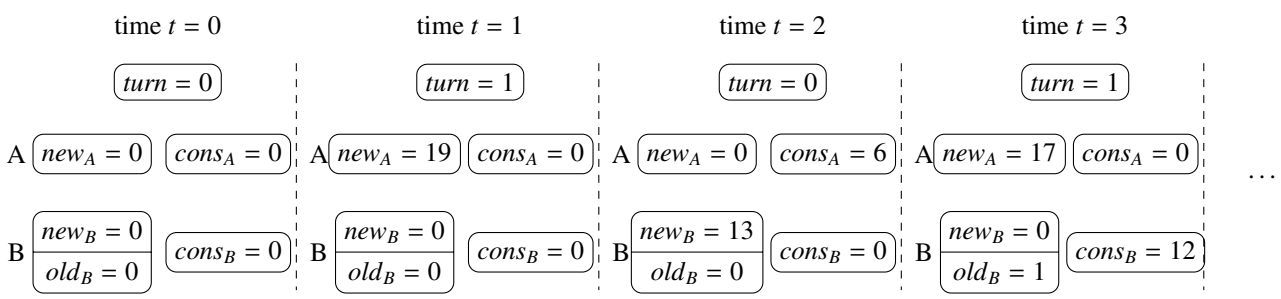


Figure 5: An example of the first states for a possible sequences in CP_3.

Context properties for CP_3

static properties $Posi_x$	$(\!\lvert Posi_x \rvert\!)$	$\triangleq$	$\Box(x \geqslant 0)$ for $x \in X'_{\mathrm{CP}}$
initial property $Init_x$	$(\!\lvert Init_x \rvert\!)$	$\triangleq$	$x = 0$ for $x \in X'_{\mathrm{CP}}$
initial property Lim_A	$(\!\lvert Lim_A \rvert\!)$	$\triangleq$	$\Box(new_A \leqslant \mathrm{PL})$
initial property Lim_B	$(\!\lvert Lim_B \rvert\!)$	$\triangleq$	$\Box(new_B + old_B \leqslant \mathrm{PL})$
transition property Old_B	$(\!\lvert Old_B \rvert\!)$	$\triangleq$	$\Box \bigwedge_{k=0}^{\mathrm{PL}}((turn = 0 \wedge new_B + old_B = k) \rightarrow \mathsf{X}(old_B + cons_B = k))$
transition property New_B	$(\!\lvert New_B \rvert\!)$	$\triangleq$	$\Box \bigwedge_{k=0}^{\mathrm{PL}}((turn = 1 \wedge new_A = k) \rightarrow \mathsf{X}(new_B + cons_A = k))$
transition property $Turn_0$	$(\!\lvert Turn_0 \rvert\!)$	$\triangleq$	$\Box((turn = 0 \rightarrow \mathsf{X}(turn = 1))$
transition property $Turn_1$	$(\!\lvert Turn_1 \rvert\!)$	$\triangleq$	$\Box((turn = 1 \rightarrow \mathsf{X}(turn = 0))$

Capabilities for the agents in CP_3:

Capability	Owner	enabCond	*window.dom*	*window.next*
provide	*prov*	$turn = 0$	(new_A)	$new_A \leqslant 20$
$\{get_A^k\}_{1 \leqslant k \leqslant \mathrm{PL}}$	A	$turn = 1 \wedge new_A = k$	$(cons_A)$	$cons_A \leqslant k$
$\{get_B^k\}_{1 \leqslant k \leqslant \mathrm{PL}}$	B	$turn = 0 \wedge new_B + old_B = k$	$(cons_B)$	$cons_B \leqslant k$

We now comment on the differences between these definitions and those from examples CP_1 and CP_2:

- Property schemes $Posi_x$ and $Init_x$ are unchanged, except that their scope no longer contains old_A and now contains *turn*. Note that $Init_{turn}$ ensures that the alternance of values for variable *turn* starts by a state with $turn = 0$, thus the first transition of any allowed execution is a b-transition.
- Property Old_B now only constrains b-transitions.
- Property Old_A does not hold anymore: in CP_1 and CP_2, it specifies the evolution of variable old_A, which does not occur in CP_3. Since the remaining resources are transferred to agent B, a new property New_B replaces Old_A: it ensures that, at each a-transition, new_B is set by the quantity left by A after she consumes some of the resource.
- Properties $Turn_0$ and $Turn_1$ ensures the alternance between $a-$ and $b-$transitions: variable *turn* behaves as a bit which is swapped at each transition.

Now, notice the capabilities for agents in CP_3. There are two main differences w.r.t. the capabilities in CP_2:

- The effectiveness of each capability depends on the turn of the current state. Hence, its enabCond.
- Since A no longer stores any resource in old_A, at each state s.t. it is her turn, the quantity of resources it is able to consume is bounded by the value new_A instead of the sum $new_A + old_A$.

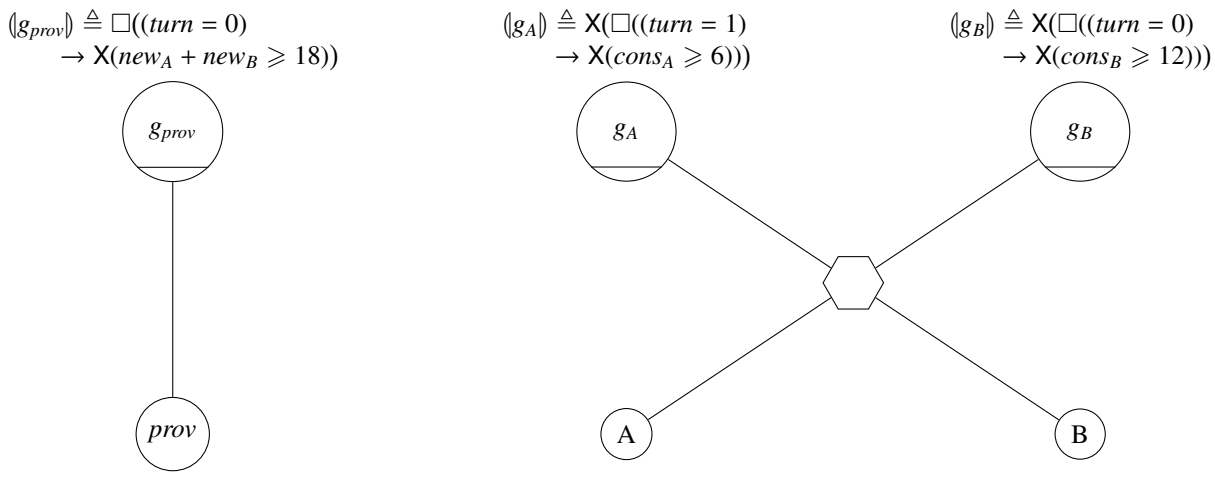


Figure 6: Goals for CP_3 and assignment $\mathcal{A}_3$ (see Example 7).

The goals and assignment for this version of our example are represented in Fig. 6. Note that g_A (resp. g_B) now only constrains a-transitions (resp. b-transitions).

In this example, whatever *prov* does, if by doing so it ensures the satisfaction of g_{prov} then it provides at least 18 units of the resource per time unit, enabling A and B to ensure the satisfaction of g_A and g_B. We say that g_{prov} *globally contributes* to g_A and g_B and we write $Contr_{\mathcal{A}_3}(g_{prov}, \{g_A, g_B\})$. ■

Again, one can also define, similarly, a relation of *local contribution*, that we do not detail here.

4. USL

In this section we introduce the formal framework that we use to check the correctness criteria for Kore assignment relation. Basically, given a specification K conforming to the Kore framework, checking a criterion consists in knowing whether goals in K are assigned to coalitions of agents able to ensure them.

Our approach consists in reducing such a question to a model-checking problem: *does a model $\mathcal{G}$ (in the logical sense) satisfy a formula φ?* where: (1) the model of the possible behaviors of the system is derived from the description of agents and context properties; and (2) the formula expresses that some coalition(s) is (are) able to ensure some goal(s).

To achieve this, a logic that allows to reason about the ability of agents to ensure temporal properties is required. This is the aim of *temporal multi-agent logics*, such as ATL (Alur et al., 2002), Strategy Logic (SL) (Mogavero et al., 2014) or USL (Updatable Strategy Logic) (Chareton et al., 2013, 2015b), which is strictly more expressive than the former two although it still enjoys a decidable model checking, and which we originally proposed to address such issues.

One of the main specific features of USL is that it enables us to express situations where agents may be part of several different interacting coalitions. Indeed, agents in USL can compose their behavior according to the different goals assigned to these coalitions. For this reason, we rely on USL in the following.

This section introduces USL. First, we recall the standard definitions regarding the semantic framework used for interpretation of multi-agent logics, namely *concurrent game structures*. Second, we define the syntax of USL. Then we introduce a number of useful technical definitions before defining the formal semantics of the logic. In all these steps, we follow most of the structure and notations of SL (Mogavero et al., 2014).

4.1. Semantic Framework

The semantic framework used to interpret USL formulas is that of *concurrent game structures*, introduced in (Alur et al., 2002) and then subsequently used with slight modifications in numerous works (Brihaye et al., 2009; Da Costa

Lopes et al., 2010; Mogavero et al., 2014). As USL builds upon SL, our definition for CGSs is the one from (Mogavero et al., 2014) (the main difference between this and the one from (Alur et al., 2002) lies in the cardinalities of the sets of states and actions: here they are enumerable whereas they are finite in (Alur et al., 2002)).

Intuitively, concurrent game structures are an extension of labelled transition systems dedicated to modelling multi-agent systems. In these systems, transitions are decided by *actions* a set of *agents* perform. At each state of any execution, each agent plays an action and the transition is determined by these actions and the current state.

Definition 7 (Concurrent Game Structure)**.** A *concurrent game structure* is a tuple $\mathcal{G} = \langle Ag, St, At, v, Act, tr, s_o \rangle$ where:

- Ag is a finite non-empty set of agents
- St is an enumerable non-empty set of *states*
- At is a finite non-empty set of *atomic propositions*
- $v : St \to \mathcal{P}(At)$ is a valuation function which maps each state s to the set of propositions true at s
- Act is an enumerable non-empty set of actions
- Let us write $Dec = Act^{Ag}$ for the set of *decisions*, *i.e.* maps from agents to actions representing the choices of an action for every agent. Then $tr : St \times Dec \to St$ is the *transition function* which maps a state and a decision to a state.
- $s_0 \in St$ is an *initial state*

Now, we define *tracks* and *paths* as non-empty finite and infinite, legal, sequences of states in a CGS.

Definition 8 (Track, Path, Execution)**.** Let $\mathcal{G} = \langle Ag, St, At, v, Act, tr, s_o \rangle$ be a CGS. A *track* in $\mathcal{G}$ is a non-empty finite sequence of states $\theta \in St^+$ s.t. for every $i \in [0, |\theta| - 1[$, there is a decision $\delta \in Dec$ s.t. $\theta_{i+1} = tr(\theta_i, \delta)$.

A *path* in $\mathcal{G}$ is an infinite sequence of states $\eta \in St^\omega$ s.t. every finite prefix of η is a track in $\mathcal{G}$.

The set of all tracks in $\mathcal{G}$ is written $Track_\mathcal{G}$ (the index will be omitted in case there is no ambiguity). An *execution* in $\mathcal{G}$ is a track or a path in $\mathcal{G}$.

Now we can define the notion of multi-strategy (the use of this word is inspired by (Bouyer et al., 2011)). A multi-strategy indicates a set of actions that may be performed by an agent depending on the history of states already met. Notice that, as usual in the literature, a multi-strategy is not attached to a single agent as multiple agents may follow the same strategy.

Note that, contrary to a number of formalisms, we allow our strategies to be non-deterministic. In practice, this approach typically encompasses situations where strategies are under-specified.

In practice, a multi-strategy which is not explicitly specified for a given track is considered to yield the set Act of all actions.

Definition 9 (Multi-strategy)**.** Let $\mathcal{G} = \langle Ag, St, At, v, Act, tr, s_o \rangle$ be a CGS. A multi-strategy ς in $\mathcal{G}$ is a map which, given a track, yields a non-empty set of actions, *i.e.* $\varsigma : Track \to \mathcal{P}_{>0}(Act)$.

The set of multi-strategies in $\mathcal{G}$ is written $MStrat_\mathcal{G}$ (the index will be omitted in case there is no ambiguity).

We now define the notion of *multi-strategy translation* along a track θ, which represents the way a multi-strategy is to be "updated" to take into account the fact that θ is now added to the history of a game.

Definition 10 (Multi-strategy Translation)**.** Let $\mathcal{G} = \langle Ag, St, At, v, Act, tr, s_o \rangle$ be a CGS, ς be a multi-strategy and θ be a track in $\mathcal{G}$. We call *translation of* ς *along* θ the multi-strategy ς^θ s.t. for any track θ', $\varsigma^\theta(\theta') = \varsigma(\theta \cdot \theta'_{\geqslant 1})$.

4.2. Syntax

The syntax of USL makes a distinction between state and path formulas. We first describe a notion of pseudo-formulas and then define formulas as pseudo-formulas where every quantified multi-strategy variable is fresh w.r.t. the scope in which it is introduced.

Definition 11 (Pseudo-formulas)**.** Let Ag be a set of agents, At a set of propositions, and X a set of (multi-strategy) variables. Then the set of *USL pseudo-formulas* is defined by the following grammar:

- *State pseudo-formulas:*
$$\psi ::= p \mid \neg\psi \mid \psi \wedge \psi \mid \langle\langle x \rangle\rangle \psi \mid (A \rhd x)\varphi \mid (A \not\rhd x)\varphi$$
- *Path pseudo-formulas:*
$$\varphi ::= \psi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi$$

where $p \in At$, $x \in X$ and A is a *coalition* (*i.e.* $A \subseteq Ag$).

The operator $\langle\langle x \rangle\rangle$ is an existencial quantifier (over multi-strategies), $(A \rhd x)$ is called a *binder* of agents in A to the multi-strategy instanciating x and $(A \not\rhd x)$ is called an *unbinder* of agents in A to the same multi-strategy x. The intuitive meaning for these operators in USL is given in Sect. 4.3.3, together with the formal definition for the satisfaction relation. We also refer the reader to (Chareton et al., 2015b) for a more detailed introduction to USL and its different operators.

Definition 12 (Sub-pseudo-formulas)**.** The set $Sf(\varphi)$ of *sub-pseudo-formulas* of a pseudo-formula φ is defined by induction on φ:

- if $\varphi = p \in At$, then $Sf(\varphi) = \{\varphi\}$
- if $\varphi = \neg\psi, \varphi = \langle\langle x \rangle\rangle\psi, \varphi = (A \rhd x)\psi, \varphi = (A \not\rhd x)\psi$ or $\varphi = \mathsf{X}\psi$ then $Sf(\varphi) = \{\varphi\} \cup Sf(\psi)$
- if $\varphi = \psi_1 \wedge \psi_2$ or $\varphi = \psi_1 \mathsf{U} \psi_2$ then $Sf(\varphi) = \{\varphi\} \cup Sf(\psi_1) \cup Sf(\psi_2)$

As multi-strategy variable *names* are taken into account in the semantics of formulas, some care must be taken when a quantifier is encountered. Thus, well-formed formulas are pseudo-formulas such that every quantifier introduces a fresh multi-strategy variable w.r.t. the scope in which it appears.

Definition 13 ((Well-Formed) formula , subformula)**.** A pseudo-formula φ is a *(well-formed) formula* if for any sub-pseudo-formula $\langle\langle x \rangle\rangle\varphi'$ of φ and every sub-pseudo-formula $\langle\langle y \rangle\rangle\varphi''$ of φ', x and y are distinct variables. Given a formula φ, the set of *subformulas* of φ is defined as its set of sub-pseudo-formulas and is also denoted by $Sf(\varphi)$.

Definition 14 (Free Variables)**.** The set of *free variables* of a formula φ, written $\mathrm{FV}(\varphi)$, is defined by induction on φ:

- $\mathrm{FV}(p) = \emptyset$, for $p \in At$
- $\mathrm{FV}(\neg\varphi) = \mathrm{FV}(\mathsf{X}\varphi) = \mathrm{FV}(\varphi)$
- $\mathrm{FV}(\varphi_1 \wedge \varphi_2) = \mathrm{FV}(\varphi_1 \mathsf{U} \varphi_2) = \mathrm{FV}(\varphi_1) \cup \mathrm{FV}(\varphi_2)$
- $\mathrm{FV}(\langle\langle x \rangle\rangle\varphi) = \mathrm{FV}(\varphi) \setminus \{x\}$
- $\mathrm{FV}((A \rhd x)\varphi) = \mathrm{FV}((A \not\rhd x)\varphi) = \mathrm{FV}(\varphi) \cup \{x\}$

Notation 2. The *length* of a formula φ is defined as $|Sf(\varphi)|$ and is written $|\varphi|$.

Definition 15 (Sentence)**.** A formula φ is called a *sentence* if it is *closed,* that is if $\mathrm{FV}(\varphi) = \emptyset$.

4.3. Semantics

We now give a number of technical definitions that are required to define the notion of satisfaction between a CGS and a formula. We first define the *evaluation contexts* that are used for USL, and different operations on these contexts (Sect. 4.3.1). In Sect. 4.3.2 we introduce the set of *outcomes* of an *evaluation context* and a state. Then we give the definition of satisfaction in Sect. 4.3.3.

4.3.1. Evaluation Contexts

As in SL or ATL$_{sc}$, USL relies on a notion of "environment" (called "assignment" in SL (Mogavero et al., 2014)) to evaluate the subformulas of a sentence. However, contrary to SL, we make a distinction between two parts, namely *assignments*, that keep track of multi-strategy variable instantiations, and *commitments*, that store the bindings of agents to multi-strategy variables. This distinction is important because USL allows to *compose* multi-strategies so that a given agent may be committed to *several* multi-strategies at once.

Definition 16 (Assignment, Commitment, Context)**.**

An *assignment* α is a dictionary which associates a multi-strategy with every multi-strategy variable in its domain of definition, *i.e.* a partial function $\alpha : X \nrightarrow \mathit{MStrat}$.

A *commitment* γ gathers bindings between a set of agents and variables they are bound to, *i.e.* it is a relation $\gamma \subseteq Ag \times X$.

A *context* κ is a "well-formed" pair $\langle \alpha, \gamma \rangle$ of an assignment α and a commitment γ, *i.e.* a pair s.t. each multi-strategy variable associated with an agent in the commitment is instantiated: $\gamma(Ag) \subseteq \mathrm{dom}(\alpha)$.

Notation 3. We write $\alpha_\emptyset$ for the empty assignment (s.t. $\mathrm{dom}(\alpha_\emptyset) = \emptyset$), $\gamma_\emptyset$ for the empty commitment (s.t. $\mathrm{dom}(\gamma_\emptyset) = \emptyset$), and $\kappa_\emptyset$ for the empty context (s.t. $\kappa_\emptyset = \langle \alpha_\emptyset, \gamma_\emptyset \rangle$).

The notion of multi-strategy translation is now extended to assignments, which will be useful to define the notion of outcome in the next section. A translated assignment is one where every multi-strategy instance is itself translated.

Definition 17 (Assignment Translation, Context Translation)**.** Let $\mathcal{G} = \langle Ag, St, At, v, Act, tr, s_o \rangle$ be a CGS, $\kappa = \langle \alpha, \gamma \rangle$ be a context and θ be a track in $\mathcal{G}$. We call *translation of α along θ* the assignment α^θ s.t. $\mathrm{dom}(\alpha^\theta) = \mathrm{dom}(\alpha)$ and for any $x \in \mathrm{dom}(\alpha^\theta)$, $\alpha^\theta(x) = (\alpha(x))^\theta$. The translation for the context κ is simply defined as $\langle \alpha, \gamma \rangle^\theta = \langle \alpha^\theta, \gamma \rangle$.

During the semantic evaluation of a formula, the context must be transformed as we encounter a multi-strategy quantifier, a binding operator or an unbinding operator. For a quantifier $\langle\!\langle x \rangle\!\rangle$, we simply update the current assignment for x. For a binding operator $(A \rhd x)$, the commitment must be updated with a pair $\langle a, x \rangle$ for every agent a in A. Finally, for an unbinding operator $(A \not\rhd x)$, all pairs $\langle a, x \rangle$ must be removed from the commitment, for any agent a in A.

Definition 18 (Context Transformations)**.** Let $\kappa = \langle \alpha, \gamma \rangle$ be a context, $A \subseteq Ag$ be a coalition, x be a multi-strategy variable and ς be a multi-strategy. We define the transformations $[x \mapsto \varsigma]$ on assignments and $[A \oplus x]$ and $[A \ominus x]$ on commitments as follows:

$$\alpha[x \mapsto \varsigma](y) = \begin{cases} \varsigma & \text{if } x = y \\ \alpha(y) & \text{otherwise} \end{cases}$$

$$\gamma[A \oplus x] = \gamma \cup \{\langle a, x \rangle \mid a \in A\}$$

$$\gamma[A \ominus x] = \gamma \setminus \{\langle a, x \rangle \mid a \in A\}$$

This notation is extended to contexts: $\kappa[x \mapsto \varsigma] = \langle \alpha[x \mapsto \varsigma], \gamma \rangle$, $\kappa[A \oplus x] = \langle \alpha, \gamma[A \oplus x] \rangle$ and $\kappa[A \ominus x] = \langle \alpha, \gamma[A \ominus x] \rangle$.

Remark 2. Note that applying any of these three transformations to a context gives a context as result. Indeed, they all preserve the satisfaction of the *well-formed* constraint for contexts introduced in Def. 16.

4.3.2. Outcomes

In SL (Mogavero et al., 2014), a number of definitions ("s-total" strategies and assignments, "complete" assignments...) is introduced to help characterising a play, that is the unique outcome of a game. Here, the situation is a bit different owing to non-determinism of our multi-strategies and, instead of a unique play, we will end up with a set of *outcomes*. More precisely, a context induces an *outcome* function that maps every track θ to a set of executions that can happen if agents, starting from θ, play according to the multi-strategies stored in κ.

To define the outcome function, we must first define the set of possible immediate *successors* of a given track in a context $\kappa = \langle \alpha, \gamma \rangle$.

Definition 19 (Successor Function)**.** Let $\mathcal{G} = \langle Ag, St, At, v, Act, tr, s_o\rangle$ be a CGS and $\kappa = \langle \alpha, \gamma\rangle$ be a context. Then the *successor* function $succ_\kappa : Track \to \mathcal{P}(St)$ induced by κ characterizes , for any track θ, the set of transitions that are possible if each agent respects the different multi-strategies it is bound to:

$$succ_\kappa(\theta) = \{tr(\text{last}(\theta), \delta) \mid \forall \langle a, x\rangle \in \gamma \cdot \delta(a) \in \alpha(x)(\theta)\}$$

Remark 3 (Finite Executions). Note that, in USL, as the successor function may yield the empty set, some executions may be *finite.* We use a common interpretation of temporal operators over finite executions, as introduced in Sect. 2.1. The semantics of the temporal operator X is less regular than in the usual framework with infinite executions: in particular, it does not commute with $\neg$. Nevertheless, it enables the framework to encompass cases where an agent is committed to contradictory multi-strategies, that is multi-strategies which indicate disjoint sets of actions after a given track. Detecting such cases brings significant additional expressive power, especially through the expression of equality between multi-strategies. The details are given in (Chareton et al., 2015b)

We finally come to the definition of *outcomes* of a context and a track, which is the set of *finite* and infinite executions (*i.e.* tracks and paths) that are possible when the history of the game is given by the track.

Definition 20 (Outcomes)**.** Let $\mathcal{G} = \langle Ag, St, At, v, Act, tr, s_o\rangle$ be a CGS, θ be a track and $\kappa = \langle \alpha, \gamma\rangle$ be a context. We define the set of *outcomes* of κ and θ in $\mathcal{G}$ as the set $out(\kappa, \theta)$ of tracks and paths λ, in $\mathcal{G}$, s.t. $\lambda_0 = \text{last}(\theta)$ and:

- if λ is a path: for any $i \in \mathbb{N}$, $\lambda_{i+1} \in succ_{\kappa^{\lambda_{\leqslant i}}}(\lambda_{\leqslant i})$
- if λ is a track $\lambda_{\leqslant n}$:
  - for any $i < n$, $\lambda_{i+1} \in succ_{\kappa^{\lambda_{\leqslant i}}}(\lambda_{\leqslant i})$
  - and $succ_{\kappa^\lambda}(\lambda) = \emptyset$.

4.3.3. Satisfaction

Definition 21 (Satisfaction)**.** Let $\mathcal{G} = \langle Ag, St, At, v, Act, tr, s_o\rangle$ be a CGS and κ be a context. Then the *satisfaction relation* $\models$ is defined by induction on formulas, for every state $s \in St$ and executions λ in $\mathcal{G}$, as follows:

- State formulas
  - $\mathcal{G}, \kappa, s \models p$ iff $p \in v(s)$, with $p \in At$
  - $\mathcal{G}, \kappa, s \models \neg\psi$ iff $\mathcal{G}, \kappa, s \not\models \psi$
  - $\mathcal{G}, \kappa, s \models \psi_1 \wedge \psi_2$ iff $\mathcal{G}, \kappa, s \models \psi_1$ and $\mathcal{G}, \kappa, s \models \psi_2$
  - $\mathcal{G}, \kappa, s \models \langle\!\langle x\rangle\!\rangle\psi$ iff there is a multi-strategy $\varsigma \in MStrat_\mathcal{G}$ s.t. $\mathcal{G}, \kappa[x \mapsto \varsigma], s \models \psi$
  - $\mathcal{G}, \kappa, s \models (A \rhd x)\varphi$ iff for any $\lambda \in out(\kappa[A \oplus x], s), \mathcal{G}, \kappa[A \oplus x], \lambda \models \varphi$
  - $\mathcal{G}, \kappa, s \models (A \not\rhd x)\varphi$ iff for any $\lambda \in out(\kappa[A \ominus x], s), \mathcal{G}, \kappa[A \ominus x], \lambda \models \varphi$
- Path formulas
  - $\mathcal{G}, \kappa, \lambda \models \psi$ iff $\mathcal{G}, \kappa, \lambda_0 \models \psi$, if ψ is a state formula
  - $\mathcal{G}, \kappa, \lambda \models \neg\varphi$ iff $\mathcal{G}, \kappa, \lambda \not\models \varphi$
  - $\mathcal{G}, \kappa, \lambda \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{G}, \kappa, \lambda \models \varphi_1$ and $\mathcal{G}, \kappa, \lambda \models \varphi_2$
  - $\mathcal{G}, \kappa, \lambda \models \mathsf{X}\varphi$ iff $|\lambda| > 1$ and $\mathcal{G}, \kappa^{\lambda_0\lambda_1}, \lambda_{\geqslant 1} \models \varphi$
  - $\mathcal{G}, \kappa, \lambda \models \varphi_1 \mathsf{U} \varphi_2$ iff there is a number $i \in \mathbb{N}$ s.t. $|\lambda| > i$, s.t. $\mathcal{G}, \kappa^{\lambda_{\leqslant i}}, \lambda_{\geqslant i} \models \varphi_2$ and s.t. for any $0 \leqslant j < i$, $\mathcal{G}, \kappa^{\lambda_{\leqslant j}}, \lambda_{\geqslant j} \models \varphi_1$.

Given the empty context $\kappa_\emptyset$ and a sentence ψ, we write $\mathcal{G}, s \models \psi$ iff $\mathcal{G}, \kappa_\emptyset, s \models \psi$, and $\mathcal{G} \models \psi$ iff $\mathcal{G}, s_0 \models \psi$.

Let us comment a bit on this definition:

- The operator $\langle\langle x \rangle\rangle$ is an existential quantifier over multi-strategies: a formula $\langle\langle x \rangle\rangle \varphi$ is true in a state s of a CGS $\mathcal{G}$, under context κ, iff there is a multi-strategy ς s.t. the formula φ is true in s and $\mathcal{G}$ under the context κ enriched by the fact that x is instantiated by ς.
- The operator $(A \rhd x)$ is a binding of agents in A to the multi-strategy instantiating x in the current context: a formula $(A \rhd x)\psi$ is true in a state s of a CGS $\mathcal{G}$, under context κ, iff the formula ψ is true in any execution in the outcomes of s and $\kappa[A \oplus x]$, where $\kappa[A \oplus x]$ is the context κ enriched with the fact that agents in A are now bound to the multi-strategy instantiating x in κ (in addition to the multi-strategies they were already bound to in κ, if there are some). In USL, this set of outcomes may be empty (if there are agents bound to multi-strategies in empty intersection in the context). In such a case, the current execution stops.
- $(A \not\rhd x)$ unbinds agents in A from the multi-strategy instantiating x in the current context: it is interpreted in a similar way as $(A \rhd x)$, except that the binding of agents in A to x is deleted from the current context (instead of being added).
- The semantics of temporal operators follows the classical definition of their interpretation in possibly finite executions introduced in Sect. 2.1 for LTL_{Kore}.

5. Construction of a CGS out of a Kore model

In this section, we perform the first step in the reduction of the assignment problem to a model checking problem for USL: we build a CGS out of an instance of Kore. Then, in Sect. 6, we formalize the correctness criteria as USL formulas. Then, verifying a correctness criterion comes to check whether the obtained formula is satisfied by the CGS.

From an instance K of Kore with variables in a set X, let us define a CGS, $\mathcal{G}_{\mathsf{K}} = \langle Ag, St, S_o, At, v, Act, tr \rangle$. Let us notice that we consider a set S_0 of initial states instead of a single initial state in the original definition of CGS.

Definition 22 (Agents)**.** The *set of agents* in $\mathcal{G}_{\mathsf{K}}$ is the set of agents in K:

$$Ag \triangleq \text{Agent}$$

The set of atomic propositions is defined out of all the atoms that appear in the goals, in the preconditions of capabilities and in the context properties.

Let φ be a formula in $\text{Cond}(X)$. We write $AtSf(\varphi)$ the set of its atomic subformulas, (that is the set of its subformulas of the form $t \sim t'$, where $t, t' \in \text{Iterm}(X)$ and $\sim \in \{<, >, =, \leqslant, \geqslant\}$, see Def. 4). This notation extends to a set Γ of formulas: $AtSf(\Gamma) = \bigcup_{\varphi \in \Gamma} AtSf(\varphi)$.

Definition 23 (Atomic propositions)**.** Let $C = \text{Goal} \cup \text{Agent.owns.enabCond} \cup CP$, then:

$$At \triangleq AtSf(\{(\!|\varphi|\!)\}_{\varphi \in C})$$

In order to define the set of states, we need to consider the semantics of atoms in At, in the sense of the theory of integers. More precisely, given a set $s \subseteq At$ of atoms, we consider a formula in quantifier-free Presburger arithmetic (QFP), noted $\overline{s}$, which consists of the conjunction of all the atoms in s and all the negations of atoms in $At - s$. Notice that for any $s \subseteq At$ and any $p \in At$, either p is a logical consequence of $\overline{s}$ (written $\overline{s} \models_{\text{QFP}} p$) or $\{s, p\}$ is not consistent ($\overline{s} \models_{\text{QFP}} \neg p$).

Definition 24 (States)**.** A state s in St is a subset of At that satisfies the following conditions:

- $\overline{s}$ is consistent in the sense of the quantifier-free Presburger arithmetic;
- all static context properties are logical consequences of $\overline{s}$: for any $scp \in SCP$, $\overline{s} \models_{\text{QFP}} (\!|scp|\!)$.

Definition 25 (Initial states)**.** The initial states are those that entail the initial context properties.

$$S_0 \triangleq \{s \mid s \in St \text{ and for any } icp \in ICP, \overline{s} \models_{\text{QFP}} (\!|icp|\!)\}$$

The definition of the valuation function is straightforward.

Definition 26 (Valuation)**.** Let $s \in St$ and $at \in At$, then $v(s) \triangleq s$.

In Kore, actors express their choices by deciding the values of some variables. We will use these choices to define the actions of $\mathcal{G}_K$. First, we define a function that, for each agent and each state, indicates the set of *playable choices* for this agent in this state, where a choice is a set of states in St.

In state s, if a has a capability c s.t. c.enabCond holds in s, then for any tuple of values in $c.window.next$, for variables in $c.window.dom$, a can force the system to go in a state that is consistent with this tuple of values. The set of such states defines a choice for a in s.

In order to enable an agent to play one or several capabilities at the same time, the set of available choices for a at s is closed under intersection.

Finally, we consider that an agent may choose *not* to interact with the system at a given state. Doing so, she does not modify the value of any variable: we model this possibility by integrating, for each agent and each state, the set St as an available choice.

Definition 27 (Function of playable choices $ch : (Ag \times St) \to \mathcal{P}(\mathcal{P}(St))$)**.** Given an agent a, a state $s \in St$, and a set $St' \subseteq St$, $St' \in ch(a, s)$ iff

- there is a capability (enabCond, *window*) for a s.t. $s \models_{\text{QFP}} \text{enabCond}$, a valuation $v \in \mathsf{Var}(window) \to \mathbb{Z}$ s.t. $St' = \{s' \mid \overline{s'} \wedge (\!|v|\!)$ is consistent in QFP$\}$ (where, for any valuation of variables v, $(\!|v|\!)$ designates the QFP formula $\bigwedge_{x \in \text{dom}(v)} x = v(x)$).
- or there are two choices $St_1, St_2 \in ch(a, s)$ s.t. $St = St_1 \cap St_2$. In other words, $ch(a, s)$ is closed by intersection.
- or $St' = St$

The function of available choices allows us to define in a straightforward way the set Act of actions of the CGS $\mathcal{G}_K$. Notice that we keep the correspondence from an action back to the available choice through the function *Choice*.

Definition 28 (The set of actions Act and the function *Choice*)**.** Given an agent a and a state s, we consider an ordering on the set of available choices $ch(a, s)$. Let us use the following notation: $ch(a, s) = St_{a,s,1}, \ldots, St_{a,s,k_{a,s}}$. Then, let $k = \max\{k_{a,s} \mid a \in Ag, s \in St\}$. We define the set of actions in $\mathcal{G}_K$ as follows: $Act = [1, \ldots, k]$.

Now, let us define the function *Choice* as follows:

$$Choice(a, s, ac) \triangleq \begin{cases} St_{a,s,ac} & \text{if } ac \leqslant k_{a,s} \\ St_{a,s,k_{a,1}} & \text{otherwise} \end{cases}$$

Remark 4. Contrary to the set of available choices, the set Act of actions is shared among all agents at any state. In the general case, given an agent a and a state s, Act contains more elements that $ch(a, s)$. The value of $Choice(a, s, ac)$ for $ac > k_{a,s}$ is then arbitrarily chosen among available choices.

In order to define the transition function of the CGS, we need to take into account the transition context properties. Their semantics is given as a formula of the following form (see Sect. 2.2.2): $\bigwedge_{k \in I}(\varphi_k \to \mathsf{X}(x = t_k))$ where I is a finite set of indices and for all $k \in I$, $\varphi_k \in \text{Cond}$ and $t_k \in \text{Iterm}$.

Given a state s, we define the candidate successors of s as the set of states in St such that every transition context property is fulfilled from s to s'.

Definition 29 (Candidate successors for a state)**.** Given a state $s \in St$, the set of candidate successors $Csucc(s)$ for s is defined as follows:

$$Csucc(s) \triangleq \Big\{ s' \mid \text{for any } tcp \in TCP \text{ s.t.} (\!|tcp|\!) \triangleq \bigwedge_{k \in I}(\varphi_k \to \mathsf{X}(x = t_k)), \\ \text{for any } k \in I, \\ \text{if } \overline{s} \models_{QFP} \varphi_k \text{ then } \overline{s'} \models_{QFP} x = t_k \Big\}$$

Now we can define the transition function of $\mathcal{G}_{\mathsf{K}}$. Intuitively, in a state, if every agent performs an action, we consider the intersection of all the corresponding choices. The successor state belongs to this intersection and also satisfies the transition context properties. If several successor states are possible, the transition function chooses the state that is the closest to the current state (if a variable is not involved in an agent action, nor in a transition property, then its value is left unchanged).

Definition 30 (Transition function tr_f)**.** Given a state $s \in \mathcal{G}$ and a decision $\delta : Ag \to Act$, the transition function returns the successor state. It is defined as follows:

$$\mathrm{dom}(tr_f) \triangleq \left\{(s,\delta) \mid Csucc(s) \cap \bigcap_{a\in Ag} Choice(a,s,\delta(a)) \neq \emptyset\right\}$$

and for any $(s,\delta) \in \mathrm{dom}(tr_f)$,

$$tr(s,\delta) \triangleq Csucc(s) \cap \bigcap_{a\in Ag} Choice(a,s,\delta(a)) \cap P_{Pass}(s,\delta)$$

where:

$$P_{Pass}(s,\delta) \triangleq \Big\{t \mid \text{for all } p \in At,\ \text{if there are } u,v \in Csucc(s) \cap \bigcap_{a\in Ag} Choice(a,s,\delta(a)) \text{ s.t. } p \in u \text{ and } p \notin v,\ \text{then } p \in t \text{ iff } p \in s\Big\}$$

Intuitively, $P_{Pass}(s,\delta)$ is the set of states sharing with s their valuation over the set of atoms whose truth value is decided neither by the agents nor by the transition context properties.

Thus, the model $\mathcal{G}_{\mathsf{K}}$ provides an interpretation of K and paves the way for solving the assignment problem.

6. Formalization of correctness criteria

In this section, we present the formalization of the correctness criteria. In Sect. 6.1, we express (in terms of USL formulas) that coalitions of agents ensure the satisfaction of the different correctness criteria for K. In Sect. 6.2, we study the logical relations between the criteria. In Sect. 6.3, we illustrate the collaboration and contribution criteria through an example.

6.1. Definition of the criteria

We use the following notations:

- $\overrightarrow{x}$ denotes a vector of multi-strategy variables, which we call a multi-strategy *profile* variable in the remainder. Then, for any coalition of agents $A = \{a_1,\dots,a_n\}$, the notation $(A,\overrightarrow{x})$ abbreviates the sequence $(a_1,x_{a_1}),\dots(a_n,x_{a_n})$;
- $[\![\cdot]\!]$ is the universal quantifier over multi-strategy variables. In other words, $[\![x]\!]\varphi \triangleq \neg\langle\!\langle x\rangle\!\rangle\neg\varphi$.

Definition 31 (Formalization of the correctness criteria in USL)**.** Let K be an instance of Kore with assignment $\mathcal{A}$. Let G be a set of goals in K and let g be a goal in K s.t. $g \notin G$. Then:

$$\begin{aligned}
(\!|LC_{\mathcal{A}}(G)|\!) &\triangleq \textstyle\bigwedge_{g\in G} (\langle\!\langle \overrightarrow{x_g}\rangle\!\rangle(\mathcal{A}(g) \rhd \overrightarrow{x_g})(\!|g|\!)) \\
(\!|GC_{\mathcal{A}}(G)|\!) &\triangleq \langle\!\langle \overrightarrow{x}\rangle\!\rangle(\textstyle\bigwedge_{g\in G}(\mathcal{A}(g) \rhd \overrightarrow{x_g})(\!|g|\!)) \\
(\!|Coll_{\mathcal{A}}(g,G)|\!) &\triangleq \langle\!\langle \overrightarrow{x_g}\rangle\!\rangle(\mathcal{A}(g) \rhd \overrightarrow{x_g})(\!|g|\!) \wedge (\!|GC_{\mathcal{A}}(G)|\!)) \\
(\!|Contr_{\mathcal{A}}(g,G)|\!) &\triangleq \Big[\langle\!\langle \overrightarrow{y_g}\rangle\!\rangle(\mathcal{A}(g) \rhd \overrightarrow{y_g})(\!|g|\!)\Big] \wedge \Big[[\![\overrightarrow{x_g}]\!]\Big((\mathcal{A}(g) \rhd \overrightarrow{x_g})(\!|g|\!) \to [(\mathcal{A}(g) \rhd \overrightarrow{x_g})(\!|GC_{\mathcal{A}}(G)|\!)]\Big)\Big]
\end{aligned}$$

Let us comment on the formulas in Def. 31.

- The assignment is locally correct iff for each goal g, there is a multi-strategy profile s.t., by playing it, the agents to which g is assigned can ensure its satisfaction. Hence, we check the satisfaction of this criterion by considering one (possibly different) multi-strategy profile per considered goal. Let us consider the agents capabilities for CP_1, given in Ex. 4. We see that *prov* can play the multi-strategy *always*$[New_A \geqslant 6]$ (consisting in restricting to the choices of values for new_A and new_B s.t. $new_A \geqslant 6$ at any time of the execution) and, if *prov* does so, A can play *always*$[Cons_A \geqslant 6]$ so that g_A is ensured. Similarly, by playing respectively *always*$[New_B \geqslant 12]$ and *always*$[Cons_B \geqslant 12]$, *prov* and B can ensure g_B. So, $(\langle\!\langle \overrightarrow{x_{g_A}} \rangle\!\rangle(\{A, prov\} \rhd \overrightarrow{x_{g_A}})(\!|g_A|\!)) \wedge (\langle\!\langle \overrightarrow{x_{g_B}} \rangle\!\rangle(\{B, prov\} \rhd \overrightarrow{x_{g_B}})(\!|g_B|\!))$ is true: $\mathcal{G}_{CP_1} \models_{\text{USL}} (\!|LC_{\mathcal{A}_1}(\{g_A, g_B\})|\!)$.

- For the global correctness, we consider one single multi-strategy profile, which imposes on each agent to act in a coherent way. The assignment is globally correct iff there is a multi-strategy profile $\overrightarrow{x}$ s.t. for each goal g, if the agents in the coalition $\mathcal{A}(g)$ play according to $\overrightarrow{x}$ (in the definition, we note $\overrightarrow{x_g}$ the part of $\overrightarrow{x}$ that concerns the agents in $\mathcal{A}(g)$), then they ensure the satisfaction of g. We can easily see that $(\!|GC_{\mathcal{A}_1}(G)|\!)$ is not true in $\mathcal{G}_{CP_1}$. Indeed, *prov* cannot play a multi-strategy that satisfies both g_A and g_B at the same time. (According to its capabilities, *prov* cannot deliver more than 15 units of the resource at a time).

- To ensure that a goal g globally collaborates to a set of goals G, we need a multi-strategy profile $\overrightarrow{x}$ s.t., if followed by the agents in $\mathcal{A}(g)$, $\overrightarrow{x}$ ensures at the same time that:

  - g is satisfied
  - the evolution of the model is constrained in such a way that the assignment $\mathcal{A}$ becomes globally correct for the set of goals G.

  According to this definition, g_{prov} globally collaborates to g_A and g_B in CP_2 (see Sect. 3.3). Indeed, since it may produce up to 20 units of the resource per time unit, *prov* can play a multi-strategy that will allow both A and B to ensure their respective goal. This point is formally proved in Ex. 8. Furthermore, recall that $\mathcal{A}_2(g_A)$ is reduced to $\{A\}$ and $\mathcal{A}_2(g_B)$ is reduced to $\{B\}$ and observe that CP_2 is not globally correct: to be able to ensure their goals, A and B depend on the multi-strategy played by *prov*.

- The contribution relation is a universally quantified variant of the collaboration : a goal g globally contributes to a set of goals G iff

  - the agents in $\mathcal{A}(g)$ are able to ensure g,
  - for any multi-strategy profile $\overrightarrow{x_g}$ that makes $\mathcal{A}(g)$ ensure g, $\overrightarrow{x_g}$ also makes $\mathcal{A}(g)$ constrain the evolution of the system in such a way that G becomes globally correct.

  In CP_2, by playing, for example, the multi-strategy consisting in setting new_A to 0 and new_B to 16, *prov* ensures its goal of producing at least 16 units per time unit, but it prevents A and B to ensure both goals g_A and g_B. This point is also formally proved in Ex. 8. On the other hand, in CP_3, the satisfaction of g_{prov} (providing at least 18 units of the resource) by the provider allows A an B to ensure their goals, provided the new assignment of both g_A and g_B to $\{A, B\}$. So $(\!|Contr_{\mathcal{A}_3}(g_{prov}, \{g_A, g_B\})|\!)$ is true in $\mathcal{G}_{CP_3}$.

Thus, for any correctness criterion C from Sect. 3, and for any instance K of Kore, the satisfaction of C by K is formalized by the relation $\mathcal{G}_{\mathsf{K}} \models_{\text{USL}} (\!|C|\!)$.

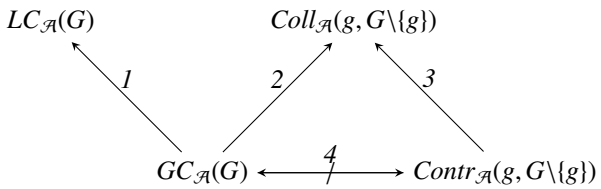
6.2. Relations between the criteria

The following theorem exposes the logical relations between our criteria.

Theorem 1. *The possible entailment relations between our correctness criteria are given in the following figure, where arrows* 1, 2 *and* 3 *represent a strict entailment relation, and the crossed out arrow* 4 *means that there is no entailment between* $GC_{\mathcal{A}}(G)$ *and* $Contr_{\mathcal{A}}(g, G\backslash\{g\})$*, in either direction. Each arrow should be read by universally quantifying the assignment* $\mathcal{A}$*, the set of goals* G *and any goal* $g \in G$*. For example:*

- ***Entailment*** *For any instance* K *of* Kore *with assignment* $\mathcal{A}$ *and for any subset* G *of goals in* K*, for any* $g \in G$*, if* $\mathcal{G}_K \models_{USL} (\!|GC_{\mathcal{A}}(G)|\!)$ *then* $\mathcal{G}_K \models_{USL} (\!|Coll_{\mathcal{A}}(g, G\backslash\{g\})|\!)$.

- ***Strictness*** *It is not true that for any instance* K *of* Kore *with assignment* $\mathcal{A}$ *and for any subset* G *of goals in* K*, for any* $g \in G$*, if* $\mathcal{G}_K \models_{USL} \langle\!|Coll_{\mathcal{A}}(g, G\backslash\{g\})|\!\rangle$ *then* $\mathcal{G}_K \models_{USL} \langle\!|GC_{\mathcal{A}}(G)|\!\rangle$.



Proof sketch. Each item in this proof sketch refers to the arrow in figure above that has the corresponding label.

1. Entailment : straightforward. Strictness: as seen in the comments on Def. 31 above, CP_1 provides a counterexample.
2. Entailment: suppose there is a general multi-strategy profile $\overrightarrow{x}$ s.t., by playing it, every coalition ensures its goal. Then, by playing along $\overrightarrow{x}$, the agents in g ensure at the same time the satisfaction of g and the global correction of the model reduced to $G \setminus \{g\}$. Strictness: as seen in the comments on Def. 31 above, CP_2 is a counterexample for the converse.
3. Entailment : straightforward. Strictness: CP_2 provides a counterexample, as detailed in Ex. 8.
4. Left to right: CP_3 satisfies $Contr_{\mathcal{A}_3}(g_{prov}, GC(\{g_A, g_B\}))$ but, to be able to ensure their goals, A and B depend on the satisfaction of g_{prov} by *prov*, so $GC_{\mathcal{A}_3}(\{g_A, g_B\})$ is not true. Right to left: consider a minimal example where *provider* is able to provide 5 units of the resource per time unit, and is assigned both goals g_- to provide 2 units, and g_+ to provide 4 units. This model is globally correct but g_- does not contribute to $\{g_+\}$. □

6.3. Illustrating example

To illustrate the formalization of the assignment problem and the definitions for our correctness criteria given in Def. 31, let us focus on example CP_2.

Example 8 (CP_2, collaboration and contribution)**.** We investigate the correctness of the assignment in CP_2 according to the criteria of

- global collaboration of g_{prov} for $\{g_A, g_B\}$,
- global contribution of g_{prov} for $\{g_A, g_B\}$,

In the following propositions, we state that the assignment in CP_2 is correct according to the first of these criteria and is not correct according to the second.

Proposition 1. *CP_2 satisfies the global collaboration of g_{prov} for $\{g_A, g_B\}$ Let $\mathcal{G}_{CP,2}$ be the model derived from CP_2 thanks to the procedure described in Sect. 5. Now, from Def. 31 and the semantics for goals in CP_2 given in Fig. 4, we can express the following formalization of this proposition:*

$$\mathcal{G}_{CP,2} \models Coll_{\mathcal{A}_2}(g_{prov}, \{g_A, g_B\})$$

where

$$Coll_{\mathcal{A}_2}(g_{prov}, \{g_A, g_B\}) \triangleq \langle\!\langle x \rangle\!\rangle (prov \rhd x)\Big((\Box \mathsf{X}(new_A + new_B \geqslant 16)) \wedge \langle\!|GC_{\mathcal{A}_2}(\{g_A, g_B\})|\!\rangle\Big) \tag{1}$$

and

$$\langle\!|GC_{\mathcal{A}_2}(\{g_A, g_B\})|\!\rangle \triangleq \langle\!\langle y \rangle\!\rangle \langle\!\langle z \rangle\!\rangle \Big(\Big((A \rhd y)\mathsf{XX}(\Box(cons_A \geqslant 6))\Big) \wedge \Big((B \rhd z)\mathsf{XX}(\Box(cons_B \geqslant 12))\Big)\Big) \tag{2}$$

The proof is given in Appendix A.

Proposition 2. *CP_2 does not satisfy the global contribution of g_{prov} for $\{g_A, g_B\}$. Let again $\mathcal{G}_{CP,2}$ be the model derived from CP_2 thanks to the procedure described in Sect. 5. The formalization of this proposition is*

$$\mathcal{G}_{CP,2} \models Contr_{\mathcal{A}_2}(g_{prov}, \{g_A, g_B\})$$

where

$$Contr_{\mathcal{A}_2}(g_{prov}, \{g_A, g_B\}) \triangleq \langle\!\langle x \rangle\!\rangle (prov \rhd x)(\Box \mathsf{X}(new_A + new_B \geqslant 16)) \\ \wedge \Big([\![x]\!] \big((prov \rhd x)(\Box \mathsf{X}(new_A + new_B \geqslant 16)) \rightarrow ((prov \rhd x)(\|GC_{\mathcal{A}_2}(\{g_A, g_B\})\|)) \big) \Big), \quad (3)$$

The proof is given in Appendix B. ■

7. Related Work

This work was initially developed in the context of Requirements Engineering (Chareton et al., 2011) and took inspiration from the state-of-the-art in this domain, in particular from Kaos (van Lamsweerde, 2009; Letier and Van Lamsweerde, 2002b; Letier, 2002). In this method, goals are gradually refined until reaching so-called requirements. Then, agents are assigned the responsibility of realizing the latter by relying on operations (our agents are directly assigned goals to simplify the presentation). In some developments of Kaos, a notion of controllable and monitorable conditions (Letier and Van Lamsweerde, 2002a; van Lamsweerde, 2004) is used as a criterion of satisfiability of realization: an agent can perform an operation if it monitors the variables in its pre-conditions and controls the ones in its post-conditions. So, in Kaos and contrary to Kore, (1) capabilities are not conditioned by the state of the system; and (2) agents interactions are not analyzed.

Another important RE approach is Tropos (Chopra et al., 2010a,b; Mallya and Singh, 2006; Chopra and Singh, 2009). In this line of work, a notion of *role* is introduced which gathers a set of specifications to be satisfied by the system. The two notions of agents and roles are then confronted. The adequacy between them is examined using propositional logic: roles are described through commitments and agents may ensure these depending on their capabilities. Considerations on time and interactions between agents are only led using natural language. Thus the method makes the verification of questions of the sort: "can agent a ensure transition tr?" possible. But the possible interactions between agents, as modifications of the common environment, are not considered formally. Kore precisely aims at unifying the multi-agent and behavioral aspects.

On the logical side, ATL and ATL* (Alur et al., 2002) consider the absolute ability of coalitions to ensure propositions whatever other agents do. But there is no contextualization w.r.t. the strategies followed by different agents. More recently, this contextualization was considered for ATL* (Brihaye et al., 2009). This proposition only uses implicit quantification over strategies for coalitions, preventing from considering different strategy quantifications for a given coalition. This problem was also tackled in SL (Mogavero et al., 2010) where quantification over strategies is made using explicit variables. Our logic USL was first developed in (Chareton et al., 2013, 2015b). Its syntax is inspired by that of SL, but it contains in addition treatment for the composition of several multi-strategies for a given agent. This enables to formalize the different possible interactions between coalitions of agents occuring in our correctness criteria for the assignment problem in Kore.

8. Conclusion and Future Work

In this article, we proposed a framework to model the assignment of behavioral goals to agents, described with capabilities. Then, we addressed the evaluation of such an assignment, informally referred to as the *assignment problem*. We proceed in two steps:

- Rather than defining one criterion that would provide a unique "yes or no" answer, we think it is more relevant to define several correctness criteria, each involving a different level of interaction between agents. We compared these criteria through a logically defined entailment relation between them.

- We provide a formalization of the different criteria using a temporal multi-agent logic, USL, and we reduce the verification of these criteria to the model-checking problem of this logic.

As a future work, the relevance of new correctness criteria for the assignment could be investigated. A direction would be to develop a formal language dedicated to the specification of criteria, using the satisfaction of goals by the coalitions they are assigned to as atoms, and the relations between these goals as operators. Thanks to such a language, we could extend and refine the criteria that can be checked in Kore.

Another direction is to study dependence relations between the multi-strategies that are played by the different coalitions. In the contribution relation for example, a coalition A_1 is able to find a favorable multi-strategy profile, whatever another coalition A_2 does (because of the nesting of multi-strategy quantifiers). In other words, A_1 knows the whole multi-strategy profile chosen by A_2 when choosing its own multi-strategy profile, which is a very strong assumption. However, it is possible in USL to characterize several forms of independence of A_1's multi-strategy profile with respect to A_2's multi-strategy profile, so that this question could be integrated in the definition of new correctness criteria.

In (Chareton et al., 2015b), we proved that the model-checking problem for USL is decidable, but does not support any elementary bound. Nevertheless, we have a strong conjecture stating that the restriction of USL to memoryless multi-strategies is decidable in pspace (Chareton, 2014). Thus, restricting to memoryless multi-strategies appears as an important condition for a tractable use of our proposition. Then, research should be led in order to further characterize the class of systems for which a memoryless semantics is adequate. Finally, it can happen that some goals are not fully achievable by the agents they are assigned to. Especially, so called *soft goals* don't have any clear cut satisfaction criterion. Then, considering and searching the best multi-strategies with regards to these goals would rise further analyses. Different notions of optima are indeed expressible in USL and could be used for defining different notions of optimal multi-strategies.

References

Alur, R., Henzinger, T. A., Kupferman, O., 2002. Alternating-time temporal logic. J. ACM 49 (5), 672–713.

Beugnard, A., Jézéquel, J.-M., Plouzeau, N., Watkins, D., 1999. Making components contract aware. Computer 32 (7), 38–45.

Bouyer, P., Markey, N., Olschewski, J., Ummels, M., 2011. Measuring permissiveness in parity games: Mean-payoff parity games revisited. In: ATVA. pp. 135–149.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J., 2004. Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems, 203–236.

Brihaye, T., Da Costa Lopes, A., Laroussinie, F., Markey, N., 2009. ATL with strategy contexts and bounded memory. Logical Foundations of Computer Science, 92–106.

Chareton, C., 2014. Modélisation formelle d'exigences et logiques temporelles multi-agents. Ph.D. thesis, Toulouse, ISAE.

Chareton, C., Brunel, J., Chemouil, D., 2011. A formal treatment of agents, goals and operations using alternating-time temporal logic. In: Formal Methods, Foundations and Applications - 14th Brazilian Symposium, SBMF 2011, São Paulo, Brazil, September 26-30, 2011, Revised Selected Papers. pp. 188–203.
URL http://dx.doi.org/10.1007/978-3-642-25032-3_13

Chareton, C., Brunel, J., Chemouil, D., 2013. Towards an updatable strategy logic. In: Proceedings 1st International Workshop on Strategic Reasoning, SR 2013, Rome, Italy, March 16-17, 2013. pp. 91–98.
URL http://dx.doi.org/10.4204/EPTCS.112.14

Chareton, C., Brunel, J., Chemouil, D., 2015a. Evaluating the assignment of behavioral goals to coalitions of agents. In: Formal Methods: Foundations and Applications - 18th Brazilian Symposium, SBMF 2015, Belo Horizonte, Brazil, September 21-22, 2015, Proceedings. Vol. 9526 of Lecture Notes in Computer Science. Springer International Publishing, pp. 56–73.
URL http://dx.doi.org/10.1007/978-3-319-29473-5_4

Chareton, C., Brunel, J., Chemouil, D., 2015b. A logic with revocable and refinable strategies. Inf. Comput. 242, 157–182.
URL http://dx.doi.org/10.1016/j.ic.2015.03.015

Chopra, A., Dalpiaz, F., Giorgini, P., Mylopoulos, J., 2010a. Modeling and reasoning about service-oriented applications via goals and commitments. In: Advanced Information Systems Engineering. Springer, pp. 113–128.

Chopra, A., Dalpiaz, F., Giorgini, P., Mylopoulos, J., 2010b. Reasoning about agents and protocols via goals and commitments. In: Proc. of the 9th Intl Conf. on Autonomous Agents and Multiagent Systems-Volume 1. pp. 457–464.

Chopra, A., Singh, M., 2009. Multiagent commitment alignment. In: Proc. of The 8th Intl Conf. on Autonomous Agents and Multiagent Systems-Volume 2. pp. 937–944.

Da Costa Lopes, A., Laroussinie, F., Markey, N., 2010. ATL with strategy contexts: Expressiveness and model checking. In: IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). Vol. 8. pp. 120–132.

De Giacomo, G., Vardi, M. Y., 2013. Linear temporal logic and linear dynamic logic on finite traces. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. IJCAI '13. AAAI Press, pp. 854–860.
URL http://dl.acm.org/citation.cfm?id=2540128.2540252

Letier, E., 2002. Reasoning about agents in goal-oriented requirements engineering. Ph.D. thesis, Université Catholique de Louvain.
Letier, E., Van Lamsweerde, A., 2002a. Agent-based tactics for goal-oriented requirements elaboration. In: Proc. of the 24th Intl Conf. on Software Engineering. ACM, pp. 83–93.
Letier, E., Van Lamsweerde, A., 2002b. Deriving operational software specifications from system goals. In: Proc. of the 10th ACM SIGSOFT Symposium on Foundations of Software Engineering. ACM, p. 128.
Maier, M. W., 1998. Architecting principles for systems-of-systems. Systems Engineering 1 (4), 267–284.
Mallya, A., Singh, M., 2006. Incorporating commitment protocols into Tropos. Agent-Oriented Software Engineering VI, 69–80.
Manna, Z., Pnueli, A., 1995. Temporal verification of reactive systems - safety. Springer.
Mogavero, F., Murano, A., Perelli, G., Vardi, M. Y., 2014. Reasoning about strategies: On the model-checking problem. ACM Transactions on Computational Logic (TOCL) 15 (4), 34.
Mogavero, F., Murano, A., Vardi, M. Y., 2010. Reasoning about strategies. In: IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). Vol. 8. pp. 133–144.
Szyperski, C., 2002. Component Software: Beyond Object-Oriented Programming, 2nd Edition. Addison-Wesley.
van Lamsweerde, A., 2003. From system goals to software architecture. In: Formal Methods for Software Architectures. pp. 25–43.
van Lamsweerde, A., 2004. Elaborating security requirements by construction of intentional anti-models. In: ICSE. pp. 148–157.
van Lamsweerde, A., 2009. Requirements Engineering - From System Goals to UML Models to Software Specifications. Wiley.
Yu, E. S.-K., 1996. Modelling strategic relationships for process reengineering. Ph.D. thesis, University of Toronto, Toronto, Ont., Canada, Canada, uMI Order No. GAXNN-02887 (Canadian dissertation).
Yu, E. S. K., 2009. Social modeling and *i**. In: Conceptual Modeling: Foundations and Applications. pp. 99–121.

Appendix A. Proof of Prop. 1

CP_2 satisfies the global collaboration of g_{prov} for $\{g_A, g_B\}$

We have that g_{prov} globally collaborates to $\{g_A, g_B\}$ iff

$$\mathcal{G}_{CP,2} \models Coll_{\mathcal{A}_2}(g_{prov}, \{g_A, g_B\})$$

Thanks to Def. 21, this is iff there is a multi-strategy $\varsigma \in MStrat_{\mathcal{G}_{CP,2}}$ s.t.

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto \varsigma\rangle\rangle, \gamma_\emptyset\rangle, s_0 \models (prov \rhd x)\Big((\Box \mathsf{X}(new_A + new_B \geqslant 16)) \wedge (\!|GC_{\mathcal{A}_2}(\{g_A, g_B\})|\!)\Big) \tag{A.1}$$

where s_0 is any state in $\mathcal{G}_{CP,2}$ satisfying the set of initial context properties $\{Init_x\}_{x \in X_{CP,2}}$ defined in Ex. 3, that is any state in which the value for every variable in $X_{CP,2}$ is equal to 0. Now, let us look at *prov*'s capability in Ex. 6: she has capability *provide*. Its enabCond is always satisfied and at any time it enables *prov* to fix the values of new_A and new_B in any way s.t. $new_A + new_B \leqslant 20$.

In particular, let us consider the partial valuation v over $\{new_A, new_B\}$ defined by $v(new_A) = 7$ and $v(new_B) = 13$. Thanks to Def. 27, we know that in any state s in $\mathcal{G}_{CP,2}$, *prov* has a choice $C = \{s' \mid \{\overline{s'}, (\!|v|\!)\}$ is consistent in QFP$\}$. And thanks to Def. 30, we know that if *prov* plays this choice from any state s, then the next state is in $C \cap Csucc(s)$, where $Csucc(s)$ is the set of states s' of $\mathcal{G}_{CP,2}$ s.t. for any $c \in \{A, B\}$, $\overline{s'}$ is consistent with formula $old_c = k - cons_c$ where $s \models_{QFP} k = new_c + old_c$ (see the definition for context properties Old_c, in Ex. 3). One easily cheks that this intersection is not empty.

For any state s in $\mathcal{G}_{CP,2}$, let us write $ac(s)$ any action s.t. $Choice(a, s, ac(s)) = C$ (see Def. 28).

Now, we define the multi-strategy $always[new_A = 7, new_B = 13]$ by: for any track θ in $\mathcal{G}_{CP,2}$, $\varsigma_p(\theta) = ac(\text{last}(\theta))$ and we claim that instanciationg ς by $always[new_A = 7, new_B = 13]$ in Eq. A.1 makes it true. That is:

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto always[new_A = 7, new_B = 13]\rangle\rangle, \gamma_\emptyset\rangle, s_0 \models (prov \rhd x)\Big((\Box \mathsf{X}(new_A + new_B \geqslant 16)) \wedge (\!|GC_{\mathcal{A}_2}(\{g_A, g_B\})|\!)\Big) \tag{A.2}$$

This is true iff

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto always[new_A = 7, new_B = 13]\rangle\rangle, \gamma_\emptyset\rangle, s_0 \models (prov \rhd x)(\Box \mathsf{X}(new_A + new_B \geqslant 16)) \tag{A.3}$$

and

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto always[new_A = 7, new_B = 13]\rangle\rangle, \gamma_\emptyset\rangle, s_0 \models (prov \rhd x)(\!|GC_{\mathcal{A}_2}(\{g_A, g_B\})|\!), \tag{A.4}$$

which is true iff, for any $\lambda \in out(\langle\langle x \mapsto always[new_A = 7, new_B = 13]\rangle\rangle, \langle(prov, x)\rangle\rangle, s_0)$,

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto always[new_A = 7, new_B = 13]\rangle\rangle, \langle(prov, x)\rangle\rangle, \lambda \models \Box \mathsf{X}(new_A + new_B \geqslant 16) \tag{A.5}$$

and

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto always[new_A = 7, new_B = 13]\rangle, \langle(prov, x)\rangle\rangle, \lambda \models \langle\!|GC_{\mathcal{A}_2}(\{g_A, g_B\})|\!\rangle \tag{A.6}$$

Let λ be a path in $out(\langle\langle x \mapsto always[new_A = 7, new_B = 13]\rangle, \langle(prov, x)\rangle\rangle, s_0)$. By definition of $always[new_A = 7, new_B = 13]$, it is s.t. for any non-initial state λ_k of λ, $\overline{\lambda_k}$ is consistent with $v(new_A) = 7 \wedge v(new_B) = 13$. Now, let us notice that formula $new'_A + new'_B \geqslant 16$ is an atomic subformula of g_{prov}. Then it is in At and either $\overline{\lambda_k} \models_{QFP} new'_A + new'_B \geqslant 16$ or $\overline{\lambda_k} \models_{QFP} \neg(new'_A + new'_B \geqslant 16)$. Since $\overline{\lambda_k}$ is consistent with $v(new_A) = 7 \wedge v(new_B) = 13$, it cannot be that $\overline{\lambda_k} \models_{QFP} \neg(new'_A + new'_B \geqslant 16)$. Then $new'_A + new'_B \geqslant 16$ is true in any non-initial state λ_k of λ , which makes true Eq. A.5 and Eq. A.3. Now, Eq. A.6 is true iff there are multi-strategies ς' and ς'' s.t. for any $\lambda \in out(\langle\langle x \mapsto always[new_A = 7, new_B = 13], y \mapsto \varsigma', z \mapsto \varsigma''\rangle, \langle(prov, x), (A, y), (B, z)\rangle\rangle, s_0)$,

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto always[new_A = 7, new_B = 13], y \mapsto \varsigma', z \mapsto \varsigma''\rangle, \langle(prov, x), (A, y), (B, z)\rangle\rangle, \lambda \models \mathsf{XX}(\Box(cons_A \geqslant 6)) \tag{A.7}$$

and

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto always[new_A = 7, new_B = 13], y \mapsto \varsigma', z \mapsto \varsigma''\rangle, \langle(prov, x), (A, y), (B' z)\rangle\rangle, \lambda \models \mathsf{XX}(\Box(cons_B \geqslant 12)), \tag{A.8}$$

For Eq. A.7, we instanciate ς' by any multi-strategy $afterOneTimeSlot[cons_A = 7]$ s.t. for any track θ s.t. $|\theta| > 1$, $afterOneTimeSlot[cons_A = 7](\theta)$ is an action ac' s.t. $Choice(A, \text{last}(\theta), ac') = \{s \mid \{\overline{s}, cons_A = 7\}$ is consistent in QFP$\}$. Indeed, one easily checks that $\{s \mid \{\overline{s}, cons_A = 7\}$ is consistent in QFP$\}$ is an available choice for A in any state λ_k occurring as a non initial state of any $\lambda \in out(\langle\langle x \mapsto always[new_A = 7, new_B = 13], y \mapsto \varsigma', z \mapsto \varsigma''\rangle, \langle(prov, x), (A, y), (B' z)\rangle\rangle, s_0)$. Then, for any such λ_k, it is not the case that $\overline{\lambda_k} \models_{QFP} \neg(cons_A \geqslant 6)$. So $\overline{\lambda_k} \models_{QFP} \neg(cons_A \geqslant 6)$, since formula $(cons_A \geqslant 6)$ is in At.

Similarly, for Eq. A.8, we instanciate ς'' by any multi-strategy $afterOneTimeSlot[cons_B = 13]$ s.t. for any track θ s.t. $|\theta| > 1$, $afterOneTimeSlot[cons_B = 13](\theta)$ is an action ac'' s.t. $Choice(B, \text{last}(\theta), ac'') = \{s \mid \{\overline{s}, cons_B = 13\}$ is consistent in QFP$\}$.

Appendix B. Proof of Prop. 2

CP_2 does not satisfy the global contribution of g_{prov} for $\{g_A, g_B\}$

Let again $\mathcal{G}_{CP,2}$ be the model derived from CP_2 thanks to the procedure described in Sect. 5. We have that g_{prov} globally collaborates to $\{g_A, g_B\}$ iff

$$\mathcal{G}_{CP,2} \models Contr_{\mathcal{A}_2}(g_{prov}, \{g_A, g_B\})$$

that is true iff

$$\mathcal{G}_{CP,2} \models \langle\!\langle x \rangle\!\rangle (prov \rhd x)(\Box \mathsf{X}(new_A + new_B \geqslant 16)) \wedge \Big([\![x]\!] \Big((prov \rhd x)(\Box \mathsf{X}(new_A + new_B \geqslant 16)) \rightarrow ((prov \rhd x)(\langle\!|GC_{\mathcal{A}_2}(\{g_A, g_B\})|\!\rangle)) \Big) \Big), \tag{B.1}$$

which is true iff

$$\mathcal{G}_{CP,2} \models \langle\!\langle x \rangle\!\rangle (prov \rhd x)(\Box \mathsf{X}(new_A + new_B \geqslant 16)) \tag{B.2}$$

and

$$\mathcal{G}_{CP,2} \models [\![x]\!] \Big((prov \rhd x)(\Box \mathsf{X}(new_A + new_B \geqslant 16)) \rightarrow ((prov \rhd x)(\langle\!|GC_{\mathcal{A}_2}(\{g_A, g_B\})|\!\rangle)) \Big) \tag{B.3}$$

From the development in Appendix A, we know that Eq. B.2 is true. Now, we want to show that Eq. B.3 is false. To do so, we need to exhibit a multi-strategy ς s.t.

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto \varsigma\rangle, \langle(prov, x)\rangle\rangle, s_0 \not\models [\![x]\!] \Big((prov \rhd x)(\Box \mathsf{X}(new_A + new_B \geqslant 16)) \rightarrow ((prov \rhd x)(\langle\!|GC_{\mathcal{A}_2}(\{g_A, g_B\})|\!\rangle)) \Big) \tag{B.4}$$

which means a multi-strategy $\varsigma \in MStrat_{\mathcal{G}_{CP,2}}$ s.t.

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto \varsigma\rangle\rangle, \langle\langle (prov, x)\rangle\rangle, s_0 \models \Box \mathsf{X}(new_A + new_B \geqslant 16) \tag{B.5}$$

is true and

$$\mathcal{G}_{CP,2}, \langle\langle x \mapsto \varsigma\rangle\rangle, \langle\langle (prov, x)\rangle\rangle, s_0 \models \langle\!\langle GC_{\mathcal{A}_2}(\{g_A, g_B\})\rangle\!\rangle \tag{B.6}$$

is false.

Intuitively, this multi-strategy must be built so as to ensure that 16 units of the resource at least is produced at each time unit without ensuring that, at any time, A gets at least 6 of them available and B gets at least 12 of them available. We make our counterexample very simple and built our multi-strategy so that A never gets any of the resource: we define the multi-strategy $always[new_A = 0, new_B = 16]$ s.t. for any track θ, $always[new_A = 0, new_B = 16](\theta)$ is an action ac s.t.

$$Choice(prov, \text{last}(\theta), ac) = \{s \mid \{\overline{s}, new_A = 0 \wedge new_B = 16\} \text{ is consistent in QFP}\}$$

Let λ be a path in $out(\langle\langle x \mapsto always[new_A = 0, new_B = 16]\rangle\rangle, \langle\langle (prov, x)\rangle\rangle, s_0)$. It is s.t., for any non-initial state λ_k of λ, $\overline{\lambda_k}$ is consistent with $newA = 0 \wedge newB = 16$. Then it is inconsistent with atom $new_A \geqslant 6$ of g_A, which makes Eq. B.5 false.

Now, let us look at the behavior of variable old_A, fixed by transition context property Old_A from Ex. 3: it is such that old_A cannot get any strictly positive value from any state where both new_A and old_A have value 0. In addition, the behavior of $cons_A$ is fixed by the choices of A, thanks to her capability get_A from Ex. 6. This capability makes, in any state of an execution, the value of $cons_A$ bound by the value of the sum $new_A + old_A$ in the previous state. In a given path so, if neither new_A nor old_A ever gets any strictly positive value, then neither does $cons_A$. Since each variable x in the model is initialised to 0 (thanks to the respective initial context property $Init_x$ from Ex. 3), in any path $\lambda \in out(\langle\langle x \mapsto always[new_A = 0, new_B = 16]\rangle\rangle, \langle\langle (prov, x)\rangle\rangle, s_0)$, variables new_A, old_A and $cons_A$ have value 0 in every state. Then A cannot ensure the satisfaction of her goal to consume at least 6 units of the resource per time unit. So $\langle\!\langle GC_{\mathcal{A}_2}(\{g_A, g_B\})\rangle\!\rangle$ is not satisfied in the context $\langle\langle x \mapsto always[new_A = 0, new_B = 16]\rangle\rangle, \langle\langle (prov, x)\rangle\rangle$ and Eq. B.6 is false, ending our proof.

***Potential Reviewers**

* Régine Laleau
http://lacl.univ-paris12.fr/laleau/
Phone: +33.(0)1.60.74.68.40
Email: laleau@u-pec.fr

* Patrick HEYMANS
http://www.info.fundp.ac.be/~phe
Phone : +32 (0)81 72 52 75
Email: patrick.heymans@unamur.be

* Dominique Méry
http://www.loria.fr/~mery/
Phone: +33 383 59 20 19
Email: Dominique.Mery@loria.fr

LaTeX Source Files

Click here to download LaTeX Source Files: soumission-src-SCP.zip