Automated Adaptation via
Quantitative Partial Model Checking

Stefano Bistarelli, Francesco Santini
Dipartimento di Matematica e Informatica,
Università di Perugia
{bista, francesco.santini}@dni.unipg.it

Fabio Martinelli, Ilaria Matteucci
Istituto di Informatica e Telematica,
Consiglio Nazionale delle Ricerche
firstname.lastname@iit.cnr.it

ABSTRACT
We propose a formal framework to model an automated adaptation protocol based on Quantitative Partial Model Checking (QPMC). An agent seeks the collaboration of a second agent to satisfy some (fixed) condition on the actions to be executed. The provided protocol allows the two agents to automatically agree by iteratively applying QPMC.

1. INTRODUCTION
Adaptation [1] is an essential requirement for systems composed of autonomous agents acting in environments that cannot be fully known in advance. Adaptation allows agents to change their behaviour in response to changes in the environment or actions of other agents, in order, for instance, to collaboratively fulfill a given goal (i.e., goal-oriented agents). Automated adaptation let couples (or more) of intelligent agents to independently but collaboratively coordinate to fulfill a given task [8].

This paper proposes an automated adaptation procedure based on Quantitative Partial Model Checking (QPMC) [10] to suggest a formal model of a goal-oriented adaptation between two agents. In our setting, an agent A seeks the collaboration of another agent B (i.e., A and B are collaborative) to perform a set of actions with the purpose of satisfying a given goal. The requirements on these actions, which are associated with a weight, are described by φ. A receives φ from A, then chooses a part of its behaviour (e.g., B′) according to an heuristic, and finally it conveys B′ to A by moving it to φ through a QPMC function. In this way, B discloses only part of its behaviour at time. The adopted heuristics to select B′ is the minimisation of a cost k returned by QPMC(φ) = φ′, and k represents a (minimal) cost that indeed has to be paid to satisfy φ, considering the actions of B only. When A receives φ′, it can check if the overall synergy between A and B in satisfying φ is enough. If not, then A asks B to reveal different capabilities, and the procedure is repeated. It is worth noting that A and B can be set of agents (e.g., two coalitions of robots), and within each step of the adaptation procedure they select one of the internal agents according to the adopted heuristic.

The paper is structured as follows. In Sec. 2 we condense ReMs. Section 3 shows the weighted-variant of modal logic we use to define φ. Then, in Sec. 4 we introduce the QPMC function and the adaptation scheme. Section 5 summarises some of the related work and Sec. 6 concludes the paper.

2. RESIDUATED ENRICHED MONOIDS
This section introduces some notions concerning monoids enriched over semi-lattices. They allow for recasting introducing a natural approach to bipolar preferences [6] at the same time (i.e., both positive and negative).

Definition 2.1 (semi-lattices). A partial order (PO) is a triple (K, ≤, ⊥) such that K is a set of values, ≤ ⊆ K × K is a reflexive, transitive, and anti-symmetric relation and ∀a ∈ K, ⊥ ≤ a. A join semi-lattice (JSL) is a PO such that any finite, non-empty subset of K has a (obviously unique) least upper bound (lub).

Definition 2.2 (monoids). A (commutative) monoid (K, ×, 1) such that ×: K × K → K is a commutative and associative function, and ∀a ∈ K.a × 1 = a.

Definition 2.3 (ReMs). A residuated monoid (ReM) is a six-tuple (K, ≤, ⊥, ∨, ⊤, 1) s.t. (K, ≤, ⊥) is a JSL, (K, ×, 1) is a monoid, ⊤: K × K → K is a function and i) ∀a ∈ K.a × ⊥ = ⊥, and ii) ∀a, b, c ∈ K.b × c ≤ a ⇐⇒ c ≤ a b.

Residuation conveys the meaning of division.

Lemma 2.1. Let (K, ≤, ⊥, ⊤, 1) be a ReM. Then a ⊤ b is the max element of {c | b × c ≤ a}. The set is never empty, since ∀a, b ∈ K.b × 1 ≤ a.

Lemma 2.2. Let (K, ≤, ⊥, ⊤, 1) be a ReM and X ⊆ K a finite, non-empty set. Then ∀a ∈ K.a × ⊤ X = ⊤{a × x | x ∈ X}.

Note that its distributivity over ⊤ implies that × is a function that is monotone in both arguments.

Definition 2.4 (invertibility - IReM). A ReM K = (K, ≤, ⊥, ⊤, 1) is invertible (IReM) if ∀a, b ∈ K.a ≤ b ⇒ a = b × (a ⊤ b).
Formally: 

\[ [k](s) = k \in K \quad \forall s \in S \]
\[ [\phi_1 + \phi_2](s) = [\phi_1](s) + [\phi_2](s) \]
\[ [\phi_1 \times \phi_2](s) = [\phi_1](s) \times [\phi_2](s) \]
\[ [\phi_1 \text{ or } \phi_2](s) = [\phi_1](s) \text{ or } [\phi_2](s) \]
\[ [\phi](s) = \bigwedge R \{ T(s, a, s') \times [\phi](s') \} \]
\[ [a][\phi](s) = \bigwedge R \{ T(s, a, s') \times [\phi](s') \} \]

where \( R = \{ s' \in S \mid s \xrightarrow{a} s' \in T \} \)

Table 1: Semantics of w-HM. \( \biguplus \{ \emptyset \} = \bot \) and \( \bigwedge \{ \emptyset \} = 1 \).

An instantiation of IReM can be a derivation of the tropical semiring \( W_{\geq} = \langle [-n, \infty], \geq, +, \cdot, 0, \infty \rangle \), for \( \geq \) the inverse of the standard order; capped operators stand for their arithmetic variant. The \( \div \) operator, when the \( a \div b \) is \( a - b \) if positive, \( 0 \) otherwise, makes \( W_{\geq} \) invertible. An example can be \( W_{\leq} \), which also contains positive preferences \(-1, -2, \) and \(-3, \) and where \( 1 \) is \( 0 \).

3. WEIGHTED H-M LOGIC

We propose a quantitative variant of the Hennessy-Milner logic, named w-HM, to consider each transition associated with an action and a weight taken from an IReM. In Def. 3.2, we syntactically define the set \( \Phi_M \) of formulas given over an Multi Labelled Transition System (MLTS) \cite{4}.

**Definition 3.2.** A (finite) Multi Labelled Transition System (MLTS) is a five-tuple \( MLTS = (S, \text{Act}, \mathbb{K}, T, s_0) \), where \( S \) is the countable (finite) state space, \( s_0 \in S \) is the initial state, \( \text{Act} \) is a finite set of transition labels, \( \mathbb{K} \) is used for the definition of transition costs \( IReM \) \( \langle K, \leq, \times, 1, \bot \rangle \), and \( T : (S \times \text{Act} \times S) \rightarrow \mathbb{K} \) is the transition function.

**Definition 3.3.** (Syntax). Given a MLTS \( M = (S, \text{Act}, \mathbb{K}, T, s_0) \), and let \( a \in \text{Act} \), a formula \( \phi \in \Phi_M \) is syntactically expressed as follows, where \( k \in K \):

\[ \phi ::= k \mid \phi_1 \cup \phi_2 \mid \phi_1 \times \phi_2 \mid \phi_1 \ominus \phi_2 \mid (a)\phi \mid [a]\phi \]

Given, \( \mathbb{K} = \langle K, \leq, \times, 1, \bot \rangle \), we can express more than just true (corresponding to \( 1 \in K \)) and false (\( \bot \in K \)) through all the values \( k \in K \). ReM operators \( \cup, \times, \ominus \) and \( \ast \) are used in place of classical logic operators \( \lor, \land, \neg \), and \( \ast \) in order to compose the truth values of two formulas together. As a reminder, when the \( \ast \) operator is idempotent, then \( \ominus \) and \( \ominus \) coincide (see Sec. 2). Finally, we have the two classical modal operators, i.e., “possibly” \((\ast)\), and “necessarily” \([a]\).

The semantics of a formula \( \phi \) is given on a particular MLTS \( M = (S, \text{Act}, \mathbb{K}, T, s_0) \), to check the specification defined by \( \phi \) over the behaviour of a weighted transition system.

In Tab. 1 the semantics is parameterised over a state \( s \in S \), which is used to consider only the transitions that can be fired at a given step (labelled with an action \( a \)). The notion of satisfiability w.r.t. a threshold \( t \) follows.

**Definition 3.3.** (\( \models_t \)). A MLTS \( M \) satisfies a w-HM formula \( \phi \) with a threshold-value \( k \), i.e., \( M \models_t \phi \), if and only if the interpretation of \( \phi \) on \( M \) is not worse than \( t \). Formally: \( M \models_t \phi \iff t \leq [\phi](s) \).

4. AUTOMATED ADAPTATION PROTOCOL

Figure 1 depicts a possible sequence of messages exchange between two agents \( A \) and \( B \). Agent \( A \) elaborates a w-HM formula \( \phi \) to be satisfied. Then, among all the agents in range, it chooses \( B \) and sends \( \phi \) to \( B \) to ask for a collaboration aimed at the satisfaction of \( \phi \). As a syntax to describe such behaviour, we adopt a slightly different version of GPA from \cite{4} in which synchronisation is made “à la CCS” instead of “à la CSP.” Indeed, given two agent \( P \) and \( P' \), \( P \parallel P' \) describes the process in which \( P \) and \( P' \) proceed concurrently when they perform actions belonging to \( L \subseteq \text{Act} \), and independently on all the other actions.

\[ P = \{ a \mid P \} = \{ a \mid P \} \]

First of all we introduce the QPMC function with respect to the parallel composition of processes, see Tab. 2.

In Th. 4.1 a result similar to the one in \cite{2} holds.

**Theorem 4.1.** Let \( P \) and \( Q \) two processes in GPA, \( \mathbb{K} \) an IReM \( \langle K, \leq, \times, 1, \bot \rangle \) with \( k \in K \), and \( \phi \) a w-HM formula, the following holds:

\[ [\phi](P \parallel L Q) = k P \parallel [\phi](Q) \]
\[ [\phi](A \parallel L B) = k [\phi](A \parallel \phi) \]

**Step 1.** A “Receiver” agent \( B \) receives \( \phi \), selects a subset of its behaviour \( B \) (see next paragraph), and moves it into \( \phi \), thus applying \( \text{QPMC}(\phi, B) \). Then, \( B \) sends the result \( \langle \phi', k \rangle \), back to \( A \), the “Initiator” agent.

The details behind this step are described by the pseudo-code in Alg. 1, which implements the function \( \text{Compute}\Phi \). Given a process \( B \) described as the parallel composition of different procedures \( B' \), \( B'' \), \( \cdots \), \( B^{n} \), the algorithm picks the subset of \( B' \)'s that minimises the result \( k \) of \( \text{QPMC}(\phi, \langle y \parallel L \rangle) \) (lines 9-13 in Alg. 1), according to the preference order in the chosen IReM. As already introduced, \( k \) represents a minimal cost that indeed has to be paid in order to satisfy \( \phi \) (considering only the actions of \( B \)), and...
such heuristics tries to minimise the overall cost by minimising the $k$ due to $B$. The subset is selected in the power-set of $\{B', \ldots , B^N\}$, by restricting to all subsets of cardinality $l$ (lines 21-24 in Alg. 1). This parameter is initially set to 1, thus the first time returning only singleton subsets $\{\{B'\}, \ldots , \{B^N\}\}$. The motivation behind this parameter is that $B$ tries to satisfy $\phi$ at best by first using less actions as possible. After having tried all $l$-combinations (rejected by $A$), $B$ increments $l$ by one (e.g., with $l = 2$, all $\{B' \parallel \text{ } B^2\}, \ldots , \{B^{N-1} \parallel \text{ } B^N\}$ are checked), since $\phi$ cannot be satisfied with only smaller parts of $B$, and more concurrency needs to be considered.

Step 2. $A$ receives the couple $(\phi', k)$ from $B$ (see Fig. 1). Then $A$ calls the same function $\text{Compute} \Phi$, since each agent can be either the Initiator or the Receiver, and so they need to implement the same functionalities. As at Step 1, such function selects the subset of $A'$s that, this time (lines 14-18 in Alg. 1), minimises the evaluation of $[\phi'([])A']$. Due to the monotonicity of $\times$ (see Sec. 2), it also minimises $r = k \times [\phi'([])A']$, where $k$ has been obtained from $B$ at the previous step (1 in Fig. 1). According to Th. 4.1, this is equivalent to minimising $r = [\phi(\text{initiator}))A') \parallel \text{ } [\phi(\text{initiator}))A')$. If this value is not worse than a given threshold $t$, i.e., $r \geq t$, $\phi$ is (quantitatively) satisfied and the adaptation protocol ends. Otherwise, $A$ and $B$ continue repeating the same steps, e.g., 3 and 4 in Fig. 1 until an agreement is reached, or all possible combinations are unsuccessfully checked.

Complexity. The for loop at line 3 in Alg. 1 is repeated $O(N^l)$ times: at most the $l$-combinations in $\mathcal{P}(\{D', D'' , \ldots , D^N\})$. There exist efficient algorithms to solve this enumeration problem [12]. Note also that the whole code in Alg. 1 cannot be easily turned into a 0/1-knapsack problem, since the total cost of $D'$s in $E$ is not just $\text{QPMC}(D', \phi) \times \ldots \times \text{QPMC}(D', \phi)$, but it depends on $D' \parallel \ldots \parallel D^i$.

According to [2], the complexity of QPMC function (branch at lines 9-13 in Alg. 1) depends on the length of $\theta$, i.e., $|\theta|$, on the number of considered $D'$ and their dimension in terms of number of states and number of transitions; this is part of the MILTS definition of $D'$. In particular, let $n_i$ the number of states of $D^i$, and $m_i$ the number of its transitions, then the complexity of the QPMC function is $O(|\theta| \cdot \prod_{i=1}^{N} (n_i + m_i))$, re-adapting [2] on $1..N$ processes. In order to be able to end the computation, we need to assume that $N$ is finite, and that each $D'$ is a finite process, i.e., it has a finite number of states and transitions.

Note that the complexity in the second branch of Alg. 1 (lines 14-19) consists in model-checking $[\theta]_E$ over $E$. Its complexity is comparable to the one of the QPMC function, since it depends on the dimension of $\theta$ and $E$ as well.

Table 2: A QPMC function extracting a $k_P$ minimal amount of weight needed to satisfy $\phi$ (on P side).

<table>
<thead>
<tr>
<th>Algorithm 1: Selecting a behaviour of agent $D$ w.r.t. $\phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Require: Behaviour of $D = D' \parallel \ldots \parallel D^N$, an adaptation level $l$, a set $X$ of already discarded l-combinations, and a w-HM formula $\theta$</td>
</tr>
<tr>
<td>1: $\text{global} \ l \parallel \cdots \parallel$ adaptation level</td>
</tr>
<tr>
<td>2: $\text{global} \ T = \mathcal{P}({D', D'', \ldots , D^N})$ \parallel \text{Powerset}</td>
</tr>
<tr>
<td>3: $\text{global} \ X \parallel \cdots \parallel$ discarded combinations</td>
</tr>
<tr>
<td>4: $\underline{\text{function Compute} \Phi(D, \theta)}$</td>
</tr>
<tr>
<td>5: $h, k = \bot, \psi = \perp, U = \emptyset$</td>
</tr>
<tr>
<td>6: for all $(T' \in T) \wedge (T' \notin X) \wedge (</td>
</tr>
<tr>
<td>7: $E = \parallel \text{initiator}(D') \text{ s.t. } D' \in T^i$</td>
</tr>
<tr>
<td>8: if (this is Receiver) then</td>
</tr>
<tr>
<td>9: $(\psi, h) = QPMC(\theta, E)$</td>
</tr>
<tr>
<td>10: if $h \geq k$ then</td>
</tr>
<tr>
<td>11: $h = h, \phi = \psi, U = T^i$</td>
</tr>
<tr>
<td>12: end if</td>
</tr>
<tr>
<td>13: else</td>
</tr>
<tr>
<td>14: $(\phi, k) = QPMC(\theta, E)$</td>
</tr>
<tr>
<td>15: if $h \geq k$ then</td>
</tr>
<tr>
<td>16: $h = h, \phi = \psi, U = T^i$</td>
</tr>
<tr>
<td>17: end if</td>
</tr>
<tr>
<td>18: end if</td>
</tr>
<tr>
<td>19: end if</td>
</tr>
<tr>
<td>20: end for</td>
</tr>
<tr>
<td>21: $X \cup U$</td>
</tr>
<tr>
<td>22: if $(</td>
</tr>
<tr>
<td>23: $l = l + 1, X = \emptyset$</td>
</tr>
<tr>
<td>24: end if</td>
</tr>
<tr>
<td>25: return $(\phi, k)$</td>
</tr>
<tr>
<td>26: end function</td>
</tr>
</tbody>
</table>

Ensure: Either $k$ is the best minimal cost extracted by QPMC from $D$ and an adaptation level $l$ (Receiver), or $k$ is the best value obtained by evaluating $[\theta]_E(D)$ on a level $l$ (Initiator).
Example. Suppose we have two processes, \( Q \) (the Initiator) and \( P \) (the Receiver), and a request \( \phi = [a]b[0] \). Let \( P \) be \( P'' \)= \( P'''\) (with \( L = \{a, b\} \)) where

\[
P'' = (a, 1).((b, 2).0 + (b, 7).0) + (a, 5).(b, -1).0
\]

\[
P''' = (a, 1).((b, -2).0 + (b, 6).0) + (a, 2).(b, 3).0
\]

(\( + \) is the classical non-deterministic choice operator), while \( Q \) is only defined by \((a, 5).(b, 2).0\). Therefore, when \( P \) receives \( \phi \), it computes both \( \phi_1/\phi_2 \), and \( \phi_1/\phi_2 \) (loop at line 7 in Alg. 1), obtaining \( \phi_1/\phi_2 = \phi = ([a]([b]0)) \cap (1 \times [a]b) \) and \( \phi_1/\phi_2 = \phi'' = ([a]([b]0) \cap (8 \times [b]0)) \cap (6 \times [a]b) \), and accumulating \( k_{\phi_2} = 3 \) and \( k_{\phi_2} = -1 \) respectively. Since \( k_{\phi_2} \geq k_{\phi_2} \), \( P \) sends \( \phi'' \) back to \( Q \), which in turn computes \( k_{\phi_2} \times [\phi'']_Q = -1 \times ((5 \times (2 \cap (8 \times 2))) \cap (6 \times 5 \times 2)) = -1 \times (15 \cap 13) = 14 \). Note that the evaluation of \( \phi'' \) would have been instead \( k_{\phi_2} \times [\phi'']_Q = 3 \times (5 \times (2 \cap (5 \times 2))) \cap (1 \times 5 \times 2) = 3 \times (12 \cap 8) = 15 \).

Therefore, in this example the heuristic works by offering the best solution first. Note that the heuristic used by Alg. 1 is optimal (i.e., a better best solution first. Note that the heuristic used by Alg. 1 is optimal (i.e., a better best solution first. Note that the heuristic used by Alg. 1 is optimal (i.e., a better best solution first. Note that the heuristic used by Alg. 1 is optimal (i.e., a better best solution first. Note that the heuristic used by Alg. 1 is optimal (i.e., a better best solution first.

5. RELATED WORK

In [7] and [14] the authors analyse a selected list of existing design patterns for coordination of self-organising systems. In [9], Li et al. present an approach for securing distributed adaptation. A plan is synthesized and executed, allowing the different parties to apply a set of data transformations in a distributed fashion. The work in [5] provides two contributions: i) a formal framework that unifies behavioural adaptation and structural reconfiguration of components; this is used for statically reasoning whether it is possible to reconfigure a system. And ii), two cases of reconfiguration in a client/server system in which the server is substituted by another one with a different behavioural interface, and the system keeps on working transparently from the client’s point of view.

In [13] the authors focus on automated adaptation of an agent’s functionality by means of an agent factory. An agent factory is an external service that adapts agents, on the basis of a well-structured description of the software agent. Structuring an agent makes it possible to reason about an agent’s functionality on the basis of its blueprint, which includes information about its configuration.

The objective of the work in [11] is the definition of a process and tool-supported design framework to develop self-adaptive systems that consider Belief-Desire-Intention agent models as reference architectures. The authors adopt an agent-oriented approach to explicit model system goals in the requirements specification and in the system architecture design.

6. CONCLUSION

The main contribution in this paper is the use of a Partial Model Checking function as a mean to carry behavioural-information from one agent to another. We think this application deserves to be further investigated in the future, since intermediate representations seem a natural way to model partial information about agents.

We plan to improve Alg. 1 by spotting the expensive steps of a computation, and designing transformational-operators on GPA processes, with the purpose to reshape and reduce costs during adaptation procedure. Moreover, we plan to extend the protocol to more than two agents, considering the surrounding environment as well, and, finally, adapt it to constraint-based languages with time [3].

Acknowledgment. This work has been partially supported by the TENACE PRIN Project (n. 20103P34XC) and H2020-MSCA-ITN-2015-NeCS 675320.