Declarative agent control

A. Kakas[1], P. Mancarella[2], F. Sadri[3], K. Stathis[2,4], and F. Toni[2,3]

[1] Dept. of Computer Science, University of Cyprus, antonis@cs.ucy.ac.cy
[2] Dip. di Informatica, Università di Pisa, {paolo,stathis,toni}@di.unipi.it
[3] Dept. of Computing, Imperial College London, {fs,ft}@doc.ic.ac.uk
[4] School of Informatics, City University London, kostas@soi.city.ac.uk

Abstract. In this work, we extend the architecture of agents (and robots) based upon fixed, one-size-fits-all cycles of operation, by providing a framework of declarative specification of agent control. Control is given in terms of *cycle theories*, which define in a declarative way the possible alternative behaviours of agents, depending on the particular circumstances of the (perceived) external environment in which they are situated, on the internal state of the agents at the time of operation, and on the agents' behavioural profile. This form of control is adopted by the KGP model of agency and has been successfully implemented in the PROSOCS platform. We also show how, via cycle theories, we can formally verify properties of agents' behaviour, focusing on the concrete property of agents' *interruptibility*. Finally, we give some examples to show how different cycle theories give rise to different, heterogeneous agents' behaviours.

1 Introduction

To make theories of agency practical, normally a control component is proposed within concrete agent (robot) architectures. Most such architectures rely upon a fixed, one-size-fits-all cycle of control, which is forced upon the agents whatever the situation in which they operate. This kind of control has many drawbacks, and has been criticised by many (e.g. in robotics), as it does not allow us to take into account changes in the environment promptly and it does not take into account agent's preferences and "personality".

In this paper, we present an alternative approach, which models agents' control via declarative, logic-based *cycle theories*, which provide *flexible control* in that: (i) they allow the same agent to exhibit different behaviour in different circumstances (internal and external to the agent), thus extending in a non-trivial way conventional, fixed cycles of behaviour, (ii) they allow us to state and verify formal properties of agent behaviour (e.g. their interruptibility), and thus (iii) provide implementation guidelines to design suitable agents for suitable applications. Furthermore, cycle theories allow different agents to have different patterns of behaviour in the same circumstances, by varying few, well-identified components. Thus, by adopting different cycle theories we obtain behaviourally *heterogeneous* agents.

The notion of cycle theory and its use to determine the behaviour of agents can in principle be imported into any agent system, to replace conventional fixed cycles. However, in defining the cycle theory of an agent, we will assume that the agent is equipped with a pool of *state transitions* that modify its internal state. We will understand the

operation of agents simply in terms of sequences of such transitions. Such sequences can be obtained from *fixed cycles* of operation of agents as in most of the literature. Alternatively, such sequences can be obtained via fixed cycles together with the possibility of selecting amongst such fixed cycles according to some criteria e.g. the type of external environment in which the agent will operate (see the recent work of [4]). Yet another possibility, that we pursue in this paper, is to specify the required operation via more versatile cycle theories that are able to generate dynamically several cycles of operations according to the current need of the agent. This approach has been adopted in the KGP model of agency [8, 2] and implemented in the PROSOCS platform [16].

We will define a cycle theory as a logic program with priorities over rules. The rules represent possible follow-ups of (already executed) transitions. The priorities express high-level preferences of the particular agent equipped with the cycle theory, that characterise the operational behaviour of the agent, e.g. a preference in testing the preconditions of an action before it tries to execute it. We will assume that the choice for the next transition depends only on the transition that has just been executed (and the resulting state of the agent), and not on the longer history of the previous transitions. We believe this not to be restrictive, in that the effects of any earlier transitions may in any case be recorded in the internal state of the agent and reasoned upon by it. Also, the approach can be extended to take into account longer histories of transitions when deciding the next one.

2 Background

Cycle theories will be written in the general framework of Logic Programming with Priorities (LPP). Our approach does not rely on any concrete such framework. One such concrete framework could be the Logic Programming without Negation as Failure (LPwNF) [5, 9] suitably extended to deal with dynamic preferences [10]. Other concrete frameworks that could be used for LPP are, for instance, those presented in [13, 12]. Note also that our approach does not depend crucially on the use of the framework of LPP: other frameworks for the declarative specification of preference policies, e.g. Default Logic with Priorities [3], could be used instead. Note, however, that the use of a logic-based framework where priorities are encoded within the logic itself is essential, since it allows reasoning even with potentially contradictory preferences. Also, note that the choice of one logic rather than another might affect the properties of agents specified via cycle theories.

For the purposes of this paper, we will assume that an *LPP-theory*, referred to as $\mathcal{T}$, consists of four parts:

(i) a low-level part P, consisting of a logic program; each rule in P is assigned a name, which is a term; e.g., one such rule could be
$$n(X) : p(X) \leftarrow q(X, Y), r(Y)$$
with name $n(X)$;

(ii) a high-level part H, specifying conditional, dynamic priorities amongst rules in P; e.g., one such priority could be
$$h(X) : n(X) \succ m(X) \leftarrow c(X)$$

to be read: if (some instance of) the condition $c(X)$ holds, then the rule in P with name (the corresponding instance of) $n(X)$ should be given higher priority than the rule in P with name (the corresponding instance of) $m(X)$. The rule is given a name, $h(X)$;

(iii) an auxiliary part A, defining predicates occurring in the conditions of rules in P and H and not in the conclusions of any rule in P;

(iv) a notion of incompatibility which, for the purposes of this paper, can be assumed to be given as a set of rules defining the predicate $incompatible$, e.g.

$incompatible(p(X), p'(X))$

to be read: any instance of the literal $p(X)$ is incompatible with the corresponding instance of the literal $p'(X)$. We assume that incompatibility is symmetric, and refer to the set of all incompatibility rules as I.

Any concrete LPP framework is equipped with a notion of entailment, that we denote by $\models_{pr}$. Intuitively, $\mathcal{T} \models_{pr} \alpha$ iff α is the "conclusion" of a sub-theory of $P \cup A$ which is "preferred" wrt $H \cup A$ in $\mathcal{T}$ over any other any other sub-theory of $P \cup A$ that derives "conclusion" incompatible with α (wrt I). Here, we are assuming that the underlying logic programming language is equipped with a notion of "entailment" that allows to draw "conclusions". In [13, 12, 10, 9, 5], $\models_{pr}$ is defined via argumentation.

3 Abstract agent model

We assume that our agents conform to the following abstract model, which can be seen as a high-level abstraction of most agent systems in the literature. Agents are equipped with

- some *internal state*, which changes over the life-time of the agent, and is formalised in some logic-based language or via some concrete data structure in some programming language;
- some pool of *(state) transitions*, that modify the state of the agent, and may take some inputs to be "computed" or selected by
- some *selection functions* on their states.

For example, the state may consist of beliefs, desires and intentions, represented in some modal logics, as in the BDI architecture [14] and its follow-ups, e.g. [1], or commitments and commitment rules, as in [15], or beliefs, goals and capabilities, represented in concurrent logic programming, as in [7], or knowledge, goals and plan, represented in (extensions of) logic programming, as in [11].

The transitions in the given pool can be any, but, if we abstract away from existing agent architectures and models in the literature, we can see that we need at least a transition responsible for observing the environment, thus rendering the agents situated. This transition might modify the internal state differently in concrete agent architectures, to record the observed events and properties of the environment. Here, we will call such a transition *Passive Observation Introduction* (POI). POI is "passive" in the sense that, via such a transition, the agent does not look for anything special to observe, but rather it opens its "reception channel" and records any inputs what its sensors perceive. Another transition that is present in most agent systems is that of *Action Execution* (AE),

whereby actions may be "physical", communicative, or "sensing", depending on the concrete systems.

Other useful transitions besides POI and AE (see e.g. [8, 2]) may include Goal Introduction (GI), to introduce new goals into the state of the agent, taking into account changes to the state and to the external environment that somehow affect the preferences of the agent over which goals to adopt, Plan Introduction (PI), to plan for goals, Reactivity (RE), to react to perceived changes in the environment by means of condition-action/commitment-like rules, Sensing Introduction (SI), to set up sensing actions for sensing the preconditions of actions in the agent's plan, to make sure these actions are indeed executable, Active Observation Introduction (AOI), to actively seek information from the environment, State Revision (SR) to revise the state currently held by the agent, and Belief Revision (BR), e.g. by learning.

Whatever pool of transitions one might choose, and whatever their concrete specification might be, we will assume that they are represented as

$$T(S, X, S', \tau)$$

where S is the state of the agent before the transition is applied and S' the state after, X is the (possibly empty) input taken by the transition, and τ is the time of application of the transition. Note that we assume the existence of a *clock* (possibly external to the agent and shared by a number of agents), whose task is to mark the passing of time. The clock is responsible for labelling the transitions with the time at which they are applied. This time (and thus the clock) might play no role in some concrete agent architectures and models, where time is not reasoned upon explicitly. However, if the framework adopted to represent the state of the agent directly manipulates and reasons with time, the presence of a clock is required. Note also that the clock is useful (if not necessary) to label executed actions, and in particular communicative actions, to record their time of execution, as foreseen e.g. by FIPA standards for communication [6].

As far as the selection functions are concerned, we will assume that each transition T available to the agent is equipped with a selection function f_T, whose specification depends on the representation chosen for the state and on the specification of the transition itself. For example, AE is equipped with a selection function f_{AE} responsible for choosing actions to be executed. These actions may be amongst those actions in the plan (intention/commitment store) part of the state of the agent whose time has not run-out at the time of selection (and application of the transition) and belonging to a plan for some goal which has not already been achieved by other means.

In the next Section, we will see that, for fixed cycles, the role of the selection functions is exclusively to select the inputs for the appropriate transition when the turn of the transition comes up. Later, in Section 5, we will see that the role of selection functions when using cycle theories is to help decide which transition is preferred and should be applied next, as well as provide its input.

4 Fixed cycles and fixed operational trace

Both for fixed cycles and cycle theories, we will assume that the operation of an agent will start from some *initial state*. This can be seen as the state of the agent when it is created. The state then evolves via the transitions, as commended by the fixed cycle or

cycle theory. For example, the initial state of the agent could have an empty set of goals and an empty set of plans, or some designer-given goals and an empty set of plans. In the sequel, we will indicate the given initial state as S_0.

A *fixed cycle* is a fixed sequence of transitions of the form

$$T_1, \ldots, T_n$$

where each T_i, $i = 1, \ldots, n$, is a transition chosen from the given pool, and $n \geq 2$.

A fixed cycle induces a *fixed operational trace* of the agent, namely a (typically infinite) sequence of applications of transitions, of the form

$T_1(S_0, X_1, S_1, \tau_1), T_2(S_1, X_2, S_2, \tau_2), \ldots, T_n(S_{n-1}, X_n, S_n, \tau_n),$
$T_1(S_n, X_{n+1}, S_{n+1}, \tau_{n+1}), \ldots, T_n(S_{2n-1}, X_{2n}, S_{2n}, \tau_{2n}), \ldots$

where, for each $i \geq 1$, $f_{T_i}(S_{i-1}, \tau_i) = X_i$, namely, at each stage, X_i is the (possibly empty) input for the transition T_i chosen by the corresponding selection function f_{T_i}.

Then, a classical "observe-think-act" cycle (e.g. see [11]) can be represented in our approach as the fixed cycle:

$$POI, RE, PI, AE, AOI.$$

As a further example, a purely reactive agent, e.g. with its knowledge consisting of condition-action rules, can execute the cycle

$$POI, RE, AE.$$

Note that POI is interpreted here as a transition which is under the control of the agent, namely the agent decides when it is time to open its "reception channel". Below, in Section 8, we will see a different interpretation of POI as an "interrupt".

Note that, although fixed cycles such as the above are quite restrictive, they may be sufficiently appropriate in some circumstances. For example, the cycle for a purely reactive agent may be fine in an environment which is highly dynamic. An agent may then be equipped with a catalogue of fixed cycles, and a number of conditions on the environment to decide when to apply which of the given cycles. This would provide for a (limited) form of intelligent control, in the spirit of [4], paving the way toward the more sophisticated and fully declarative control via cycle theories given in the next Section.

5 Cycle theories and cycle operational trace

The role of the cycle theory is to dynamically control the sequence of the internal transitions that the agent applies in its "life". It regulates these "narratives of transitions" according to certain requirements that the designer of the agent would like to impose on the operation of the agent, but still allowing the possibility that any (or a number of) sequences of transitions can actually apply in the "life" of an agent. Thus, whereas a fixed cycle can be seen as a restrictive and rather inflexible catalogue of allowed sequences of transitions (possibly under pre-defined conditions), a cycle theory identifies *preferred patterns* of sequences of transitions. In this way a cycle theory regulates in a flexible way the operational behaviour of the agent.

Formally, a cycle theory $\mathcal{T}_{cycle}$ consists of the following parts.

– An *initial* part $\mathcal{T}_{initial}$, that determines the possible transitions that the agent could perform when it starts to operate (*initial cycle step*). More concretely, $\mathcal{T}_{initial}$ consists of rules of the form

$$*T(S_0, X) \leftarrow C(S_0, \tau, X), now(\tau)$$

sanctioning that, if the conditions C are satisfied in the initial state S_0 at the current time τ, then the initial transition should be T, applied to state S_0 and input X, if required. Note that $C(S_0, \tau, X)$ may be absent, and $\mathcal{T}_{initial}$ might simply indicate a fixed initial transition T_1.

The notation $*T(S, X)$ in the head of these rules, meaning that the transition T can be potentially chosen as the next transition, is used in order to avoid confusion with the notation $T(S, X, S', \tau)$ that we have introduced earlier to represent the actual application of the transition T.

– A *basic* part $\mathcal{T}_{basic}$ that determines the possible transitions (*cycle steps*) following other transitions, and consists of rules of the form

$$*T'(S', X') \leftarrow T(S, X, S', \tau), EC(S', \tau', X'), now(\tau')$$

which we refer to via the name $\mathcal{R}_{T|T'}(S', X')$. These rules sanction that, after the transition T has been executed, starting at time τ in the state S and ending at the current time τ' in the resulting state S', and the conditions EC evaluated in S' at τ' are satisfied, then transition T' could be the next transition to be applied in the state S' with the (possibly empty) input X', if required. The conditions EC are called *enabling conditions* as they determine when a cycle-step from the transition T to the transition T' can be applied. In addition, they determine the input X' of the next transition T'. Such inputs are determined by calls to the appropriate selection functions.

– A *behaviour* part $\mathcal{T}_{behaviour}$ that contains rules describing dynamic priorities amongst rules in $\mathcal{T}_{basic}$ and $\mathcal{T}_{initial}$. Rules in $\mathcal{T}_{behaviour}$ are of the form

$$\mathcal{R}_{T|T'}(S, X') \succ \mathcal{R}_{T|T''}(S, X'') \leftarrow BC(S, X', X'', \tau), now(\tau)$$

with $T' \neq T''$, which we will refer to via the name $\mathcal{P}^{T}_{T' \succ T''}$. Recall that $\mathcal{R}_{T|T'}(\cdot)$ and $\mathcal{R}_{T|T''}(\cdot)$ are (names of) rules in $\mathcal{T}_{basic} \cup \mathcal{T}_{initial}$. Note that, with an abuse of notation, T could be 0 in the case that one such rule is used to specify a priority over the *first* transition to take place, in other words, when the priority is over rules in $\mathcal{T}_{initial}$. These rules in $\mathcal{T}_{behaviour}$ sanction that, at the current time τ, after transition T, if the conditions BC hold, then we prefer the next transition to be T' over T'', namely doing T' has *higher priority* than doing T'', after T. The conditions BC are called *behaviour conditions* and give the behavioural profile of the agent. These conditions depend on the state of the agent after T and on the parameters chosen in the two cycle steps represented by $\mathcal{R}_{T|T'}(S, X')$ and $\mathcal{R}_{T|T''}(S, X'')$. Behaviour conditions are *heuristic* conditions, which may be defined in terms of the *heuristic selection functions*, where appropriate. For example, the heuristic action selection function may choose those actions in the agent's plan whose time is close to running out amongst those whose time has not run out.

– An *auxiliary part* including definitions for any predicates occurring in the enabling and behaviour conditions, and in particular for selection functions (including the heuristic ones, if needed).

– An *incompatibility part*, including rules stating that all different transitions are incompatible with each other and that different calls to the same transition but with different input items are incompatible with each other. These rules are facts of the form

$$incompatible(*T(S, X), *T'(S, X')) \leftarrow$$

for all T, T' such that $T \neq T'$, and of the form
$incompatible(*T(S,X), *T(S,X')) \leftarrow X \neq X'$
expressing the fact that only one transition can be chosen at a time.

Hence, $\mathcal{T}_{cycle}$ is an LPP-theory (see Section 2) where:
(i) $P = \mathcal{T}_{initial} \cup \mathcal{T}_{basic}$, and (ii) $H = \mathcal{T}_{behaviour}$.

In the sequel, we will indicate with $\mathcal{T}^0_{cycle}$ the sub-cycle theory $\mathcal{T}_{cycle} \setminus \mathcal{T}_{basic}$ and with $\mathcal{T}^s_{cycle}$ the sub-cycle theory $\mathcal{T}_{cycle} \setminus \mathcal{T}_{initial}$.

The cycle theory $\mathcal{T}_{cycle}$ of an agent is responsible for the behaviour of the agent, in that it induces a *cycle operational trace* of the agent, namely a (typically infinite) sequence of transitions
$T_1(S_0, X_1, S_1, \tau_1), \ldots, T_i(S_{i-1}, X_i, S_i, \tau_i),$
$T_{i+1}(S_i, X_{i+1}, S_{i+1}, \tau_{i+1}), \ldots$
(where each of the X_i may be empty), such that

- S_0 is the given initial state;
- for each $i \geq 1$, τ_i is given by the clock of the system, with the property that $\tau_i < \tau_{i+i}$;
- (*Initial Cycle Step*) $\mathcal{T}^0_{cycle} \wedge now(\tau_1) \models_{pr} *T_1(S_0, X_1)$;
- (*Cycle Step*) for each $i \geq 1$
  $\mathcal{T}^s_{cycle} \wedge T_i(S_{i-1}, X_i, S_i, \tau_i) \wedge now(\tau_{i+1}) \models_{pr} *T_{i+1}(S_i, X_{i+1})$
  namely each (non-final) transition in a sequence is followed by the most preferred transition, as specified by $\mathcal{T}_{cycle}$.

If, at some stage, the most preferred transition determined by $\models_{pr}$ is not unique, we choose arbitrarily one.

Note that, for simplicity, the above definition of operational trace prevents the agent from executing transitions *concurrently*. However, a first level of concurrency can be incorporated within traces, by allowing all preferred transitions to be executed at every step. For this we would only need to relax the above definition of $incompatible$ transitions to be restricted between any two transitions whose executions could interact with each other and therefore cannot be executed concurrently on the same state, e.g. the Plan Introduction and State Revision transitions. This would then allow several transitions to be chosen together as preferred next transitions and a concurrent model of operation would result by carrying out simultaneously the (non-interacting) state updates imposed by these transitions. Further possibilities of concurrency will be subject of future investigations.

In section 8 we will provide a simple extension of the notion of operational trace defined above.

6 Fixed versus flexible behaviour

Cycle theories generalise fixed cycles in that the behaviour given by a fixed operational trace can be obtained via the behaviour given by a cycle operational trace, for some special cycle theories. This is shown by the following theorem, which refers to the notions of *fixed cycle* and *fixed operational trace* introduced in Section 4.

Theorem 1. *Let $T_1, \ldots, T_n$ be a* fixed cycle*, and let f_{T_i} be a given selection function for each $i = 1, \ldots, n$. Then there exists a cycle theory $\mathcal{T}_{cycle}$ which induces a cycle operational trace identical to the fixed operational trace induced by the fixed cycle.*

Proof. The proof is by construction as follows.

- $\mathcal{T}_{initial}$ consists of the rule
  $*T_1(S_0, X) \leftarrow now(\tau)$
  i.e. the initial transition is simply T_1.
- $\mathcal{T}_{basic}$ consists of the following rules, for each i with $2 \leq i \leq n$:
  $*T_i(S', X') \leftarrow T_{i-1}(S, X, S', \tau), now(\tau'), X' = f_{T_i}(S', \tau')$.
  In addition $\mathcal{T}_{basic}$ contains the rule
  $*T_1(S', X') \leftarrow T_n(S, X, S', \tau), now(\tau'), X' = f_{T_1}(S', \tau')$.
- $\mathcal{T}_{behaviour}$ is empty.
- the auxiliary part contains the definitions of the given selection functions f_{T_i}, for each $i = 1, \ldots, n$.

The proof then easily follows by construction, since at each stage only one cycle step is enabled and no preference reasoning is required to choose the next transition to be executed. □

It is clear that there are some (many) cycle theories that cannot be mapped onto any fixed cycles, e.g. the cycle theory given in the next Section. So, providing control via cycle theories is a genuine extension of providing control via conventional fixed cycles.

7 An Example

In this Section we exemplify the flexibility afforded by cycle theories through a simple example. Assume that the pool of transitions consists of GI, PI, AE and POI, as described in Section 3. We start from the cycle theory corresponding to the fixed cycle given by POI, GI, PI, AE which is constructed as follows (see Theorem 1).
(1) $\mathcal{T}_{initial}$ with the following rule
$*POI(S_0, \{\}) \leftarrow$
namely, the only way an agent can start is through a POI.
(2) $\mathcal{T}_{basic}$ with the following rules
$*GI(S', \{\}) \leftarrow POI(S, \{\}, S', \tau)$
$*PI(S', Gs) \leftarrow GI(S, \{\}, S', \tau), Gs = f_{PI}(S', \tau'), now(\tau')$
$*AE(S', As) \leftarrow PI(S, Gs, S', \tau), As = f_{AE}(S', \tau'), now(\tau')$
$*POI(S', \{\}) \leftarrow AE(S, As, S', \tau)$
(3) $\mathcal{T}_{behaviour}$ is empty.

A first simple improvement, providing a limited form of flexibility, consists in refining the rule $\mathcal{R}_{GI|PI}(\cdot)$ by adding the condition that the set of goals to plan for, which are selected by the corresponding selection function f_{PI}, is non-empty. This amounts at modifying the second rule of $\mathcal{T}_{basic}$ by adding the condition $Gs \neq \{\}$ to its body.

Similarly, AE is an option after PI if some actions can actually be selected for execution. This amounts at modifying the the third rule of $\mathcal{T}_{basic}$ by adding the condition $As \neq \{\}$ to its body.

In this case, we should provide further options for choosing the transition to be executed after GI and PI, respectively. To adhere with the given original cycle, these rules could be simply suitable rules named by $\mathcal{R}_{GI|AE}(S', As)$, $\mathcal{R}_{GI|POI}(S', \{\})$ and $\mathcal{R}_{PI|POI}(S', \{\})$, i.e. AE and POI are also an option after GI, and POI is also an option after PI. With this choice, the standard operational trace is recovered by adding to the $\mathcal{T}_{behaviour}$ part of the cycle theory the following rules

$\mathcal{R}_{GI|PI}(S', Gs) \succ \mathcal{R}_{GI|AE}(S', As) \leftarrow$
$\mathcal{R}_{GI|AE}(S', As') \succ \mathcal{R}_{GI|POI}(S', \{\}) \leftarrow$
$\mathcal{R}_{GI|PI}(S', Gs') \succ \mathcal{R}_{GI|POI}(S', \{\}) \leftarrow$
$\mathcal{R}_{PI|AE}(S', As) \succ \mathcal{R}_{PI|POI}(S', \{\}) \leftarrow$

The first rule states that PI has to be preferred over AE as the next transition to be applied after GI, whenever both PI and AE are enabled. Similarly for the other rules.

A more interesting, proper extension of the original (fixed) cycle amounts at adding further options to the transition which can follow any given transition. Imagine for instance that we want to express the behaviour of a *punctual* or *timely* agent. This agent should always prefer executing actions if there are actions in the plan which have become *urgent*. This can be declaratively formalised by adding to the $\mathcal{T}_{basic}$ part the rules

$$*AE(S', As') \leftarrow T(S, X, S', \tau), As' = f_{AE}(S', \tau'), now(\tau')$$

for each transition T in the pool, and by adding to the $\mathcal{T}_{behaviour}$ part the following rules named $\mathcal{P}^{T}_{AE \succ T'}$:

$$\mathcal{R}_{T|AE}(S', As') \succ \mathcal{R}_{T|T'}(S', X') \leftarrow urgent(As')$$

for each transition T and $T' \neq$ AE, where $urgent$ is defined in the auxiliary part of the theory with the intuitive meaning. In the rest of this Section, we use $\mathcal{T}^{fix}_{cycle}$ to refer to the cycle theory corresponding to the fixed cycle POI, GI, PI, AE, and we use $\mathcal{T}^{ext}_{cycle}$ to refer to the extended cycle theory.

As a concrete example, consider an agent aiding a businessman who, while on a business trip, can choose amongst three possible goals: return home ($home$), read news ($news$), and recharge his laptop battery ($battery$). Let us use first the cycle theory $\mathcal{T}^{fix}_{cycle}$.

Suppose that, initially (when $now(1)$ holds), the agent's state is empty, namely the (businessman's) agent holds no plan or goal, and that the initial POI does not add anything to the current state. Then GI is performed as the next transition in the trace:

$$GI(S_0, \{\}, S_1, 1),$$

and suppose also that the application of GI generates the agent's goal (added to S_1) $G_1 = home$. This goal may come along with a time parameter and some temporal constraints associated with it, e.g. the actual goal can be represented by $(home, t) \wedge t < 20$. Due to space limitations, we intentionally omit here the details concerning temporal parameters of goals and actions, and the temporal constraints associated with them. Since the state contains a goal to be planned for, suppose that the selection function f_{PI} selects this goal, and the PI transition is applied next, producing two actions $book\_ticket$ and $take\_train$. Hence, the second transition of the trace is (when $now(3)$ holds)

$$PI(S_1, \{\}, S_2, 3)$$

where the new state S_2 contains the above actions.

Suppose now that the selection function f_{AE} selects the action $book\_ticket$ and hence that the next element of the trace is (when $now(4)$ holds)

$AE(S_2, \{book\_ticket\}, S_3, 4)$. (*)

In the original fixed cycle the next applicable transition is POI, and assume that this is performed at some current time, say 10. Hence the next element of the trace is (when $now(10)$ holds)

$POI(S_3, \{\}, S_4, 10)$. (**)

Imagine that this POI brings about the new knowledge that the laptop battery is low, suitably represented in the resulting state S_4. Then the next transition GI changes the state so that the goal *battery* is added, and then PI is performed to introduce a suitable plan to recharge the battery and so on.

Now suppose that we use $\mathcal{T}^{ext}_{cycle}$ instead and that the operational trace is identical up to the execution of the transition (*). At this point, the action *take_train* may have become *urgent*. Notice that it is likely that this same action was not urgent at time 3, when *book_ticket* was selected for execution, but has become urgent at time 10 (e.g. because the train is leaving at 11). Then, if we use $\mathcal{T}^{ext}_{cycle}$, the rule $\mathcal{P}^{AE}_{AE \succ POI}$ applies and the next element of the trace, replacing (**) above, becomes

$AE(S_3, \{take\_train\}, S'_4, 10)$.

This example shows how the use of cycle theories can lead to flexible behaviours. More flexibility may be achieved by allowing the agents to be interruptible, i.e. to be able to react to changes in the environment in which they are situated as soon as the perceive those changes. This added flexibility requires some further extensions, that we discuss in the next Section.

8 Interruptible agents

In our approach we can provide a declarative specification of *interruptible* agents, i.e. agents that are able to dynamically modify their "normal" (either fixed or cycle) operational trace when they perceive changes in the environment in which they are situated.

In order to obtain interruptibility, we will make use of the POI transition as the means by which an agent can react to an interrupt. Referring to the example of the previous Section, assume that our agent can book the ticket only through its laptop and, by the time it decides to actually book the ticket, the laptop battery has run out. Then, the action of recharging the laptop battery should be executed as soon as possible in order to (possibly) achieve the initial goal. Indeed, executing the booking action before recharging would not be feasible at all.

In order to model the environment where the agent is situated, we assume the existence of an environmental knowledge base Env that it is not directly under the control of the agent, in that the latter can only dynamically assimilate the knowledge contained in Env. This knowledge base can be seen as an abstraction of the physical (as opposed to the mental) part of the agent (its *body*) which, e.g. through its sensors, perceives changes in the environment. We assume that, besides the knowledge describing the agent's percepts, Env models a special propositional symbol, referred to as $changed\_env$ which holds as soon as the body of the agent perceives any new, relevant changes in the environment. The way we model the reaction of the agent to the changes represented by *changed_env* becoming true, is through the execution of a POI. We also

assume that the execution of a POI transition resets the truth value of $changed\_env$, so that the agent may be later alerted of further changes in the environment.

The Env knowledge base becomes now part of the knowledge that the agent uses in order to decide the next step in its operational trace. This is formally specified through the notion of *cycle-env operational trace*, which extends the notion of cycle operational trace introduced in Section 5, by replacing the definitions of *Initial Cycle Step* and *Cycle Step* by the following new definitions:
(*Initial Cycle-env Step*): $\mathcal{T}^0_{cycle} \wedge Env \wedge now(\tau_1) \models_{pr} *T_1(S_0, X_1)$;
(*Cycle-env Step*) for each $i \geq 1$

$\mathcal{T}^s_{cycle} \wedge T_i(S_{i-1}, X_i, S_i, \tau_i) \wedge Env \wedge now(\tau_{i+1})$
$\models_{pr} *T_{i+1}(S_i, X_{i+1})$

We can now define a notion of *interruptible agent* as follows. Let $\mathcal{T}_{cycle}$ be the cycle theory of the agent and let $T_1(\cdot), \ldots, T_i(\cdot), \ldots$ be a cycle operational trace of the agent. Let also $T_i(S_{i-1}, X_i, S_i, \tau_i)$ be an element of the given trace such that:

$Env \wedge now(\tau_i) \models \neg changed\_env$, and

$Env \wedge now(\tau_{i+1}) \models changed\_env$.

In other words, some changes have happened in the environment between the time of the execution of the transitions T_i and T_{i+1} in the trace. Then we say that the agent is interruptible if
$\mathcal{T}_{cycle} \wedge T_i(S_{i-1}, X_i, S_i, \tau_i) \wedge Env \wedge now(\tau_{i+1}) \models_{pr} *POI(S_i, \{\})$, i.e. as soon as the environment changes, in a cycle-env operational trace the next transition would be a POI.

It is worth noting that by interruptibility we do not mean here that the (executions of) transitions are interrupted, rather the trace is interrupted.

In order to make an agent interruptible, we need to extend both $\mathcal{T}_{basic}$ and $\mathcal{T}_{behaviour}$. In $\mathcal{T}_{basic}$, POI should be made an option after any other transition in the pool, which is achieved by adding the following rule $\mathcal{R}_{T|POI}(S, \{\})$, for any T:

$*POI(S', \{\}) \leftarrow T(S, X', S', \tau)$.

In $\mathcal{T}_{behaviour}$, the following set of rules, where T, T' are transitions with $T' \neq$ POI, express that POI should be preferred over any other transition if the environment has actually changed:

$\mathcal{R}_{T|POI}(S', \{\}) \succ \mathcal{R}_{T|T'}(S', X) \leftarrow changed\_env.$ (***)

Notice that, even if the above extensions are provided in the overall $\mathcal{T}_{cycle}$ theory, the interruptibility of the agent is still not guaranteed. For instance, $\mathcal{T}_{behaviour}$ could contain further rules which make a transition $T \neq$ POI preferable over POI even if $changed\_env$ holds. One way to achieve full interruptibility is by adding the condition $\neg changed\_env$ in the body of any rule in $\mathcal{T}_{behaviour}$ other than the rules (***) given above.

9 Patterns of behaviour

In this section we show how different patterns of operation can arise from different cycle theories aiming to capture different profiles of operational behaviour by agents. We assume the agent is equipped with a set of transitions, as in the KGP model [8, 2] (see Section 3 for an informal description of these transitions):

– POI, *Passive Observation Introduction*
– AE, *Action Execution*
– GI, *Goal Introduction*
– PI, *Plan Introduction*
– RE, *Reactivity*
– SI, *Sensing Introduction*
– AOI, *Active Observation Introduction*
– SR, *State Revision*

In Section 7 we have given a simple example of a cycle theory describing a punctual, timely agent which attempts to execute its planned actions in time. This agent was obtained by adding some specific rules to $\mathcal{T}_{behaviour}$ of a given cycle theory. The same approach can be adopted to obtain different profiles.

For example, we can define a *focused or committed* agent, which, once chosen a plan to execute, prefers to continue with this plan (refining it and/or executing parts of it) until the plan is finished or it has become invalid, at which point the agent may consider other plans or other goals. Hence transitions that relate to an existing plan have preference over transitions that relate to other plans. This profile of behaviour can be captured by the following rules added to $\mathcal{T}_{behaviour}$ of an appropriate cycle theory:

$$\mathcal{R}_{T|AE}(S, As) \succ \mathcal{R}_{T|T'}(S, X) \leftarrow same\_plan(S, As)$$

for any T and any $T' \neq AE$, and

$$\mathcal{R}_{T|PI}(S, Gs) \succ \mathcal{R}_{T|T'}(S, X) \leftarrow same\_plan(S, Gs)$$

for any T and any $T' \neq PI$. These rules state that the agent prefers to execute actions or to reduce goals from the same plan as the actions that have just been executed. Here, the behaviour conditions are defined in terms of some predicate $same\_plan$ which, intuitively, checks that the selected inputs for AE and PI, respectively, belong to the *same plan* as the actions most recently executed within the latest AE transition.

Another example of behavioral profile is the *impatient* pattern, where actions that have been tried and failed are not tried again. This can be captured by rules of the form:

$$\mathcal{R}_{T|T'}(S, \_) \succ \mathcal{R}_{T|AE}(S, As) \leftarrow failed(S, As)$$

for any T and any $T' \neq AE$. In this way, AE is given less preference than any other transition T' after any transition T. Intuitively, As are *failed* actions. As a result of this priority rule it is possible that such failed actions would remain un-tried again (unless nothing else is enabled) until they are timed out and dropped by SR.

If we want to capture a *careful* behaviour where the agent revises its state when one of its goals or actions times out (being careful not to have in its state other goals or actions that are now impossible to achieve in time) we would have in $\mathcal{T}_{behaviour}$ the rule:

$$\mathcal{R}_{T|SR}(S, \{\}) \succ \mathcal{R}_{T|T'}(S, \_) \leftarrow timed\_out(S, \tau)$$

for any T and any $T' \neq SR$. In this way, the SR transition is preferred over all other transitions, where the behaviour condition $timed\_out(S, \tau)$ succeeds if some goal or action in the state S has timed out at time τ.

10 Conclusions and ongoing work

We have presented an approach providing declarative agent control, via logic programs with priorities. Our approach share the aims of 3APL [4], to make the agent cycle programmable and the selection mechanisms explicit, but goes beyond it. Indeed, the approach of [4] can be seen as relying upon a catalogue of fixed cycles together with the possibility of selecting amongst such fixed cycles according to some criteria, whereas we drop the concept of fixed cycle completely, and replace it with fully programmable cycle theories.
Our approach allows us to achieve flexibility and adaptability in the operation of an autonomous agent. It also offers the possibility to state and verify properties of agents behaviour formally. In this paper we have exemplified the first aspect via an example, and the second aspect via the property of "interruptibility" of agents. The identification and verification of more properties is a matter for future work.
Our approach also lends itself to achieving heterogeneity in the overall operational behaviour of different agents that can be specified within the proposed framework. Indeed, an advantage of control via cycle theories is that it opens up the possibility to produce a variety of patterns of operation of agents, depending on the particular circumstances under which the transitions are executed. This variety can be increased, and many different *patterns or profiles of behaviour* can be defined by varying the cycle theory, thus allowing agents with (possibly) the same knowledge and operating in the same environment to exhibit heterogeneous behaviour, due to their different cycle theories. We have given a number of examples of profiles of behaviour. A systematic study of behaviour parameterisation (perhaps linking with Cognitive Science) is a matter for future work, as well as the comparison on how different behaviours affect the agents' individual welfare in different contexts.

Acknowledgments

This work was partially funded by the IST programme of the EC, FET under the IST-2001-32530 SOCS project, within the Global Computing proactive initiative. The last two authors were also supported by the Italian MIUR programme "Rientro dei cervelli".

References

1. R.H. Bordini, A. L. C. Bazzan, R. O. Jannone, D. M. Basso, R. M. Vicari, and V. R. Lesser. Agentspeak(XL): Efficient intention selection in BDI agents via decision-theoretic task scheduling. In C. Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Part III*, pages 1294 – 1302, Bologna, Italy, July 15–19 2002. ACM Press.
2. A. Bracciali, N. Demetriou, U. Endriss, A. Kakas, W. Lu, P. Mancarella, F. Sadri, K. Stathis, G. Terreni, and F. Toni. The KGP model of agency for GC: Computational model and prototype implementation. In *Proc. Global Computing 2004 Workshop*, LNCS. Springer Verlag, 2004.
3. G. Brewka. Reasoning with priorities in default logic. In *Proceedings of AAAI-94, pp. 940-945*, 1994.

4. M. Dastani, F. S. de Boer, F. Dignum, W. van der Hoek, M. Kroese, and J. Ch. Meyer. Programming the deliberation cycle of cognitive robots. In *Proc. of 3rd International Cognitive Robotics Workshop (CogRob2002)*, Edmonton, Alberta, Canada, 2002.
5. Y. Dimopoulos and A. C. Kakas. Logic programming without negation as failure. In *Logic Programming, Proceedings of the 1995 International Symposium, Portland, Oregon*, pages 369–384, 1995.
6. FIPA Communicative Act Library Specification, August 2001.
7. K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J. Ch. Meyer. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.
8. A. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. The KGP model of agency. In *Proceedings of ECAI 2004*, 2004.
9. A. C. Kakas, P. Mancarella, and P. M. Dung. The acceptability semantics for logic programs. In *Proceedings of the Eleventh International Conference on Logic Programming, Santa Marherita Ligure, Italy*, pages 504–519, 1994.
10. A. C. Kakas and P. Moraitis. Argumentation based decision making for autonomous agents. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, pages 883–890, Melbourne, Victoria, July 14–18 2003. ACM Press.
11. R. A. Kowalski and F. Sadri. From logic programming towards multi-agent systems. *Annals of Mathematics and Artificial Intelligence*, 25(3/4):391–419, 1999.
12. R.A. Kowalski and F. Toni. Abstract argumentation. *Artificial Intelligence and Law Journal, Special Issue on Logical Models of Argumentation*, 4:275–296, 1996.
13. H. Prakken and G. Sartor. A system for defeasible argumentation, with defeasible priorities. In *Proc. International Conference on Formal and Applied Practical Reasoning*, volume 1085 of *LNAI*, pages 510–524. Springer Verlag, 1996.
14. A. S. Rao and M. Georgeff. BDI Agents: from theory to practice. In *Proceedings of the 1st International Conference on Multiagent Systems, San Francisco, California*, pages 312–319, San Francisco, CA, June 1995.
15. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.
16. K. Stathis, A. Kakas, W. Lu, N. Demetriou, U. Endriss, and A. Bracciali. PROSOCS: A platform for programming software agents in computational logic. In *Proc. AT2AI-2004*, 2004.