SOCS

A COMPUTATIONAL LOGIC MODEL FOR THE DESCRIPTION, ANALYSIS AND VERIFICATION OF GLOBAL AND OPEN SOCIETIES OF HETEROGENEOUS COMPUTEES

IST-2001-32530

Profile related properties

Project number:	IST-2001-32530
Project acronym:	SOCS
Document type:	IN (information note)
Document distribution:	I (internal to SOCS and PO)
CEC Document number:	IST32530/UCY/016/IN/I/b1
File name:	4016-b1[profile-properties].pdf
Editor:	F. Athienitou, A. Kakas
Contributing partners:	UCY, ICSTM, DIPISA, CITY
Contributing workpackages:	WP5
Estimated person months:	N/A
Date of completion:	23 February 2005
Date of delivery to the EC:	18 March 2005
Number of pages:	62

ABSTRACT

This is a report on profile-related properties of computees, providing an annex to deliverable D13. We study various profiles of behaviour by giving formal definitions and proving certain properties for each profile.

Profile related properties

F. Athienitou[1], A. Bracciali[2], U. Endriss[3], A. Kakas[1], W. Lu[4], P. Mancarella[2], F. Sadri[3], K. Stathis[4], F. Toni[3]

[1] Dept. of Computer Science, University of Cyprus, Email: {cspgat1,antonis}@cs.ucy.ac.cy

[2] Dip. di Informatica, Università di Pisa, Email: {braccia,paolo}@di.unipi.it

[3] Dept. of Computing, Imperial College London, Email: {ue,fs,ft}@doc.ic.ac.uk

[4] School of Informatics, City University London, Email: {lue,kostas}@soi.city.ac.uk

ABSTRACT

This is a report on profile-related properties of computees, providing an annex to deliverable D13. We study various profiles of behaviour by giving formal definitions and proving certain properties for each profile.

Contents

1 Introduction

In this report we study a number of profiles of behaviour for computees by giving formal definitions and proving certain properties for each profile.

A profile is a pattern of behaviour characterized by operational traces and then realized through suitable cycle theories. For each profile we define the properties that must be satisfied by the operational traces and propose suitable cycle theories that induce such operational traces. Two different approaches are taken regarding the definition of cycle theories. The first approach defines (whenever possible) the cycle theory by setting some constraints and conditions on the basic part of the theory. The second approach does not put any additional constraints on the basic part of the theory and only expresses behavioural preferences by setting some conditions and adding some rules to the behaviour part of the theory, in an attempt to keep the basic level uniform for all computees.

For each cycle theory proposed, we prove that it induces operational traces which satisfy the properties of the given profile. In addition, we propose and prove comparison properties for the profiles in an attempt to exhibit their relative advantages and disadvantages. This study has shown the need for formally defining properties of the environment, since comparison between different profiles depends on what kind of environment the agents are situated in.

The profiles studied in this annex are the cautious, the actively cautious, the punctual, the impatient, the careful, the focussed and the objective profile.

The report proceeds as follows. In section 2 we review some notions from the KGP model and give some definitions from the *LPwNF* framework in which cycle theories are realized. In sections 3 to 9 the study of the different profiles is presented. In section 10 we discuss related work and finally in section 11 we summarize and evaluate the work presented. The normal cycle theory is given in appendix A.

2 Background

2.1 Recap on some notions underlying the KGP model

In this section we present an overview of the basic notions of the KGP model which are defined in detail in deliverables D4[4] and D8[3].

Internal state. This is a tuple $\langle KB, Goals, Plan, TCS \rangle$, where:

- KB is the knowledge base of the computee, and describes what the computee knows (or believes) of itself and the environment. KB consists of modules supporting different reasoning capabilities:
  - KB_{plan}, for Planning,
  - KB_{pre}, for the Identification of Preconditions of actions,
  - KB_{TR}, for Temporal Reasoning,
  - KB_{GD}, for Goal Decision,

- KB_{react}, for Reactivity, and
- KB_0, for holding the (dynamic) knowledge of the computee about the external world in which it is situated (including past communications).

Syntactically, KB_{plan},KB_{react} and KB_{TR} are abductive logic programs with constraint predicates KB_{pre} is a logic program, KB_{GD} is a logic program with priorities and KB_0 is a set of facts in logic programming, and it is (implicitly) included in all the other modules.

- *Goals* is the set of properties that the computee wants to achieve, each one explicitly time-stamped by a time variable. Syntactically, each goal is a pair of the form $\langle l[t], G' \rangle$ where
  - $l[t]$ is the *fluent literal* of the goal, referring to the time t;
  - G' is the *parent* of G.
- *Plan* is a set of actions scheduled in order to satisfy goals. Each is explicitly time-stamped. Syntactically, each action is a triple of the form $\langle a[t], G, C \rangle$ where
  - $a[t]$ is the *operator* of the action, referring to the execution time t;
  - G the parent goal, towards which the action contributes (i.e., the action belongs to a plan for the goal G). G may be a post-condition for A (but there may be other such post-conditions).
  - C are the *preconditions* which should hold in order for the action to take place successfully; syntactically, C is a conjunction of (timed) fluent literals.
- TCS is a set of constraint atoms (referred to as *temporal constraints*) in some given underlying constraint language with respect to some structure $\Re$ equipped with a notion of constraint satisfaction $\models_{\Re}$. Temporal constraints bound the time variables of goals and actions, thus implicitly defining when goals are expected to hold and when actions should be executed. Also, via the temporal constraints, actions are partially ordered.

Goals and actions are uniquely identified by their associated time, which is implicitly existentially quantified within the overall state. To aid revision and partial planning, *Goals* and *Plan* form a *tree*, whose root is represented by $\perp$ [1]. The tree is given implicitly by associating with each goal and action its parent (the second element in the corresponding tuple). *Top-level* goals and actions are children of the root of the tree, $\perp$.

Valuation of temporal constraints. Given a state $S = \langle KB, Goals, Plan, TCS \rangle$, we denote by $\Sigma(S)$ (or simply Σ, when S is clear from the context) the valuation:

$$\Sigma(S) = \{t = \tau \mid executed(a[t], \tau) \in KB_0\} \cup \{t = \tau \mid observed(l[t], \tau) \in KB_0\}$$

Intuitively, Σ extracts from KB_0 the instantiation of the (existentially quantified) time variables in *Plan* and *Goals*, derived from having executed (some of the) actions in *Plan* and having observed that (some of the) fluents in *Goals* hold (or do not hold).KB_0 provides a "virtual" representation of Σ.

[1] In the full model [3], we consider two trees, the first containing *non-reactive* goals and actions, whose root is represented by $\perp^{nr}$, the second containing *reactive* goals and actions. whose root is represented by $\perp^{r}$. All the top-level non-reactive goals are either assigned to the computee by its designer at birth, or they are determined by the Goal Decision capability. All the top-level reactive goals and actions are determined by the Reactivity capability.

Transitions. The state of a computee evolves by applying transition rules, which employ capabilities and the constraint satisfaction $\models_{\Re}$. The transitions are:

- Goal Introduction (GI), changing the top-level Goals, and using Goal Decision.
- Plan Introduction (PI), changing Goals and Plan, and using Planning and Introduction of Preconditions.
- Reactivity (RE), changing Goals and Plan, and using the Reactivity capability.
- Sensing Introduction (SI), changing Plan by introducing new sensing actions for checking the preconditions of actions already in Plan, and using Sensing.
- Passive Observation Introduction (POI), changing KB0 of KB by introducing unsolicited information coming from the environment, and using Sensing.
- Active Observation Introduction (AOI), changing KB0 of KB, by introducing the outcome of (actively sought) sensing actions, and using Sensing.
- Action Execution (AE), executing all types of actions, and thus changing KB0 of KB.
- State Revision (SR) [2], revising Goals and Plan, and using Temporal Reasoning and Constraint Satisfaction.

Selection functions Some of the transitions given above take, in addition to a state, some other input. Such additional inputs are selected by means of selection functions. These selection functions will play an important role in the cycle theories of computees that control its operational behaviour. The selection functions are goal selection, action selection, fluent selection and precondition selection, and they are all defined as mappings of the form

$$m : States \times TimeConstants \rightarrow Sets$$

from the set of all possible states of the computee and the set of all possible time-points to the set of all possible sets of items of interest. Depending on the concrete selection function, these items are

- actions in Plan,
- in Goals,
- fluents occurring anywhere in Goals,
- preconditions of actions in Plan.

We define two classes of selection functions. The first class, called core selection functions, are fixed in the model, while the second class, of heuristic selection functions, are variable and can be chosen differently for each computee. The core selection functions only select goals (for goal selection) and actions (for action selection), preconditions (for precondition selection) and fluents (for fluent selection) that still have a chance of "success" (that have not become invalid in some way), for example, they have not yet timed out. On the other hand, the heuristic

[2] We assume here that the two transitions of Goal Revision and Plan Revision in the original KGP model presented in [4, 3, 6] are combined into a single State Revision transition, whose definition is straightforward.

selection functions are used to complement the core selection functions in order to capture different behaviours for different computees. For example, if we want a computee which is timely, then its heuristic selection functions will select goals/actions/ fluents according to their urgency. Hence the core selection functions are used to indicate items that are viable selections, whereas the heuristic selection functions are used to decide which of all candidate items (as chosen by the core selection functions) are preferred and thus effectively selected. Items chosen by the heuristic selection functions are items that are preferable, in being selected, over those that are not.

The action selection function only selects actions that are not yet timed out, are not redundant and are still "useful" to perform. The goal selection function only selects goals that are not already satisfied or goals that have not yet timed out. The precondition selection function selects all those preconditions not yet known to hold or not to hold of those actions in Plan that would be selected by the action selection function.

Cycle theory. 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 empty, 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 *heuristic selection functions* (see [4] for details). 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

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

  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 a logic program with priorities (P, H, A, I) where:

(i) $P = \mathcal{T}_{initial} \cup \mathcal{T}_{basic}$,

(ii) $H = \mathcal{T}_{behaviour}$,

(iii) A is the auxiliary part of $\mathcal{T}_{cycle}$, and

(iv) I is the incompatibility part of $\mathcal{T}_{cycle}$.

In the sequel, the generic condition $now(\tau)$ in the cycle theory rules will be omitted and left implicit when this is not important for the specific rule. 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 $\mathcal{T}_{cycle} \setminus \mathcal{T}_{initial}$. We will also assume a notion of preferential entailment $\models_{pr}$, which the underlying formalism of logic programming with priorities is equipped with. Finally, unless otherwise specified, we will assume that all cycle theories include rules that make the computee *interruptible*, as specified in [5].

Operational trace. A cycle theory $\mathcal{T}_{cycle}$ induces an *operational trace*, 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 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 $\mathcal{T}^s_{cycle}$. If the most preferred transition determined by $\models_{pr}$ is not unique, we choose arbitrarily one.

2.2 The LPwNF framework

In this section we give two definitions from the *LPwNF* framework (presented in deliverable D8 [3]), which we will use later in order to prove certain properties.

- Let $\mathcal{T}$ be a theory and $\Delta \subseteq \mathcal{T}$. Then Δ is an **admissible** argument iff

  (i) Δ is consistent (i.e. conflict free)

  (ii) for any $\Delta' \subseteq \mathcal{T}$ if Δ' attacks Δ then Δ attacks Δ'.

- Let $\mathcal{T}$ be an $LPwNF$ theory and $\Delta, \Delta' \subseteq \mathcal{T}$. Then Δ' **attacks** Δ (or Δ' is a *counter argument* of Δ) iff there exists L, $\Delta_1 \subseteq \Delta'$ and $\Delta_2 \subseteq \Delta$ such that:

  (i) $\Delta_1 \vdash_{min} L$ and $\Delta_2 \vdash_{min} \overline{L}$

  (ii) $(\exists r' \in \Delta_1, r \in \Delta_2 \text{ s.t. } \Delta_2 \models_H h\_p(r, r')) \Rightarrow (\exists r' \in \Delta_1, r \in \Delta_2 \text{ s.t. } \Delta_1 \models_H h\_p(r', r))$,

  where $\overline{L}$ is any literal that conflicts L (e.g. $\overline{L} = \neg L$ or $incompatible(L, \overline{L})$ holds).

3 Cautious profile

In this section we study the cautious profile of behaviour. The computees adopting this profile are committed to execute actions only if they know that action preconditions hold. While the standard AE transition executes actions whose preconditions are not known to be false, here there is an attempt to avoid to pay the price of executing actions without any assurance about their preconditions.

In section 3.1 we give the informal definition of a cautious computee whereas in section 3.2 we give the characteristic feature of the cautious profile. Based on these definitions we propose two different ways of defining a cautious cycle theory. The cautious cycle theory presented in section 3.3 is defined by modifying the basic part of the cycle theory. This method is simpler and easier to implement. The cycle theories presented in section 3.4, on the other hand, set some conditions on the behaviour part of the cycle theory and put no constraint on the basic part, in an attempt to keep the basic level uniform for all computees.

Note that in this section we denote by $selected\_actions(S, X)$ and $selected\_goals(S, X)$, the action and goal selection functions respectively where S is the state and X is the set of selected actions or goals.

3.1 Informal statement of the property

The computee before doing an action checks its KB to see if the preconditions for that action hold, and hence performs only actions whose preconditions hold.

The above informal statement can be read either as *execute only the actions whose preconditions can be proved to hold*, or *before executing an action whose preconditions are not known to hold, perform a sensing introduction transition (SI) in order to check the preconditions in the environment, and wait for them to hold*. The cautious profile implements the first approach, while the actively cautious implements the second one.

3.2 Formal definition

State characteristics. The states of a cautious computee do not contain an AE transition whose selected actions have a precondition that can not be proved to hold. Here, the notion of *not be proved to hold* relies on the TR (temporal reasoning) capability in charge of skeptically proving fluents against the computee current representation of its environment, i.e. its knowledge base. (Skeptically represents certainty, in contrast to credulously as the possibility that fluents might hold at some points in time.)

Given an operational trace

$$T_0(S_0, X_0, S_1, \tau_0), \ldots T_i(S_i, X_i, S_{i+1}, \tau_i), \ldots,$$

for all the i such that $T_i = AE$, and for all the actions $\langle a[t], G, l_1[t] \wedge \cdots \wedge l_n[t] \rangle$ belonging to X_i, the set of actions selected as input for the transition, it holds

$$KB_{TR} \models_{TR}^{\tau_i} l_j[t] \quad j = 1, .., n.$$

Note that this characterisation only states that *if an action is executed, then its preconditions are known to hold*, and not that *if the preconditions are known to hold then the action will surely be executed*. The choice of not to have both the implications is justified by allowing for the possibility that only a subset of the actions with preconditions satisfied is executed. This may be useful when action execution costs are taken into account.

3.3 Cautious Cycle Theory based on the basic part

3.3.1 Cautious Cycle Theory

Cycle Theory characterisation. The informal definition of a cautious computee can be easily implemented by means of the following basic rule of the cycle theory (whenever performing AE, restrict the set of selected actions to those whose preconditions are known to hold):

$$\begin{aligned} Cautious_{T|AE}(S', Y) : *AE(S', Y) \leftarrow \quad & T(S, X, S', \tau), selected\_actions(S', X'), \\ & prec\_sat(X', Y, \tau'), now(\tau'), Y \neq \emptyset. \end{aligned}$$

where $selected\_actions/2$ implements the core action selection function and $prec\_sat/3$ selects, among the actions in X', those whose preconditions can be proved to hold at the current time τ' (by calling the TR capability). Note that, the condition $Y \neq \emptyset$ prevents an AE transition to occur with an empty input (note that this conforms to the normal cycle theory, which requires that AE is executed on non-empty input sets of actions).

	Preconditions are not accessible	Preconditions do not change	Preconditions may change
Goals do not change	$\approx$ (w.r.t. welfare) CP worse (objectively)	$\approx$	CP better
Goals change	?	?	?

Figure 1: CP and UP comparison

Moreover, it is required that the only rules defining AE in the cycle theory of a cautious computee are instances of the "schema" rule above. Reading the T variable in the rule as free implies that any transition can be followed by AE. This can be restricted by replacing the rule with a set of rules $Cautious_{T_1|AE}(S', Y), \ldots, Cautious_{T_k|AE}(S', Y)$, where each T_i is the label of one of the k transitions that can be followed by AE. The body of these k rules is an instance of the body of the above rule, where T has been suitably instantiated.

Compliance of the cycle theory with the state characteristic for a cautious computee. Given an operational trace of a cautious computee

$$T_0(S_0, X_0, S_1, \tau_0), \ldots T_i(S_i, X_i, S_{i+1}, \tau_i), \ldots,$$

for all i such that $T_i = AE$ and for all the actions $\langle a[t], G, l_1[t] \wedge \cdots \wedge l_n[t] \rangle \in X_i$, it holds

$$KB_{TR} \models^{\tau_i}_{TR} l_j[t] \quad j = 1, .., n.$$

Proof. Trivial by observing that the above condition is implemented as a precondition in the body of each basic rule of the cautious cycle theory enabling AE transitions.

3.3.2 Cautious profile features

In this section we analyse the distinguishing features of the cautious profile with respect to features of the environment where it may operate.

The cautious profile has been defined by suitably restricting the actions that can be executed, i.e. by refining the definition of the action selection function. Here, we compare the cautious profile (CP) with a profile (UP) operating with the unrestricted action selection function and the same cycle theory. We compare the two profiles against different classes of environments that induce differences in the profiles behaviours.

We first discuss the case of a not fully accessible environment, where the truth values of actions preconditions might not be known. Then we distinguish the cases of environments where the truth values of goals and preconditions that impair action execution may or may not vary independently of the computee actions (but are accessible). Clearly a different behaviour between CP and UP can arise regarding the actions whose preconditions are not satisfied. Figure 1 informally summarises the results.

Precondition truth values could not be proved to hold, due to a not fully accessible environment. Clearly, the CP profile will be stuck in the execution of all those actions whose precondition can not be proved to hold. It is worth reminding that precondition truth values must be *skeptically* proved, which, informally speaking, requires complete information about the current state of the environment as far as the preconditions of interest are concerned. On the other hand, the UP computee might achieve in the environment the desired effects of the actions it executes, hence performing "objectively" better than UP, even it is not able to prove that preconditions hold. However, in the lack of information about precondition truth values, CP will not be able to improve its individual welfare, which relies on the capability to skeptically prove achieved goals.[3] In the case that goals values vary independently of computee behaviours, the two profile can not be compared (see below).

Goal and precondition truth values are accessible and do not vary in the environment independently of the computee actions. In this case, both the CP and PU computee will not achieve the effects for enforcing which actions have been planned, for all the actions whose preconditions can not be proved to hold. Hence, until a successive State Revision (SR) transition, the behaviour of the two computees will be equivalent as far as these actions, and their expected effects, will be concerned.

The UP computee is aware of any executed action, also with unsatisfied preconditions, because of the $executed(a, t)$ information in its KB_0 (with t assigned to a ground time point). All these actions can be pruned from the plans in the state of the computee by a successive SR transition. This may happen as soon as one of the goals, whose satisfaction depends on the (missing) effects of the actions,[4] will become timed out.

The case of the CP computee is analogous, but for the fact that it has not executed actions, their execution is not represented in KB_0, and their time variable have not been instantiated. However, as soon as they, or a goal depending on them, will become timed out they can be pruned from the plans of the computee by a SR transition.

Hence the overall behaviour of the two computees can be considered substantially equivalent, in the sense that both of them will not be able to prove (for instance by means of a Sensing Introduction transition) the fluents expected as effects of the actions (under the above hypothesis that the environment does not change, if not for the behaviour of the computee).

Example 1 *A computee acting on the behalf of its owner wants to get a coffee from a coffee-machine before time 20. It knows that coffee can be obtained by putting money in the machine, provided that the machine is properly working. Initially it has money, and this will be spent if used in the machine. This knowledge can be represented, for instance, in the TR knowledge base as*

```
holds_at(have_money,0).

initiates(put_money,T,coffee) :- holds_at(machine,T).
initiates(repair,_,machine).
terminates(put_money,_,have_money).
```

[3] It is worth reminding that the *individual welfare* notion is based on a subjective view, where a goal is considered achieved when the computee is able to prove it against its representation of the world, see [**?**]

[4] Such a goal can be, for instance, any goal that is a predecessor of the action in the tree representing plans in the state of the computee.

```
precondition(put_money,have_money).
```

In order to achieve the goal $G = \langle coffee[t], \bot \rangle$*, with the temporal constraint* $t <= 20$*, the action* $\langle put\_money[t1], G, \{have\_money, machine\}\rangle$*, with the temporal constraint* $t1 <= 10$ *has been planned. Given this knowledge, the precondition that the machine is properly working, can not be skeptically proved by the TR capability, hence, while UP executes the actions (and finishes its money), CP does not. However, the goal can not be considered achieved by UP, since it cannot be skeptically proved from the updated* $KB_0 = \{executed(put\_money, 10).\}$ *Note that UP has no money anymore. CP analogously is not able to prove the goal (but it still has its money ...).*

Goal truth values are accessible and do not vary in the environment independently of the computee actions, precondition truth values may vary. This is the case in which CP may exhibit a more effective behaviour than UP, as far as actions producing their expected effects are concerned. Let us suppose that CP and UP are in the same state and UP executes actions whose preconditions are not satisfied, via an Action Execution (AE) transition. CP can choose the same transition AE, provided it is able to prove that preconditions holds for a non-empty subset of the same actions selected by UP. If this subset is empty, CP will perform a different transition. In any case, all the actions that will not be executed in this transition by CP will remain executable and in the plan, until they will become timed out. Assuming that in this interval action preconditions become true and CP performs another Action Execution transition, the effects for which the actions were planned can be achieved by CP, while they have not been achieved by UP when it has executed the same actions.

Example 2 *In the situation of Example 1, it is observed that the machine has been repaired at 15 by someone, so that, for the CP computee,* $KB_0 = \{observed(x, executed(repair, 15), 15)\}$*, (while for UP,* $KB_0 = \{executed(put\_money, 10), observed(x, executed(repair, 15), 15)\}$*). Let us suppose that at* 17*, CP executes another AE transition. Now both the preconditions for put_money hold, i.e. have_money and machine, and hence the action can be executed and it provides the desired effect. CP can achieve the effects of the action it performs, achieve in this case its goal, and hence improve its individual welfare. Differently, UP cannot (nor it will in the future since it has run out of money).*

Goal truth values are accessible and may vary in the environment. This case corresponds to the possibility that goals and action effects hold independently of the computee behaviour, i.e. the computee runs within an environment "unreliably" dynamic or influenced by the behaviour of other computees. Clearly, in this case it is not possible to compare CP and UP, given that the environment can advantage or disadvantage any of them.

The above analysis allows us to conclude:

Proposition 3.1 *Let CP be a cautious computee and UP a normal computee (in the above "unrestricted" sense). Let UP execute an AE transition, where a is an action whose preconditions can not be proved to hold. Then, provided that*

- *the action a belongs to the plan of CP, as well,*
- *the action a does not become timed out during an interval* T*,*
- *the truth values of the preconditions of a become provable within the interval* T*,*

- *CP executes an AE transition within T,*

CP will achieve the effects of a, while UP has not.

3.3.3 Discussion

Intuitive advantages. The main advantage of a cautious computee is not to waste time in doing actions whose preconditions are not guaranteed to hold. Other than saving the costs of executing the action, this attitude should avoid the overhead of re-planning because of potential action failures to yield the expected effects, while preserves the possibility of executing the action later on, as shown in Example 2, where the cautious computee exhibits a more effective behaviour than a computee ruled by an unrestricted profile. As shown, this advantage of the cautious computee holds in all those cases in which the delay in action execution, due to the lack of information about preconditions, might allow them to become true. The *actively cautious* computee tries to actively collect such information as soon as it realises its lack.

As explained, the environment, especially if dynamic and where (pre-) conditions may change rapidly, influences the effectiveness of the profile, so that it is possible to have examples where the punctual computee increases its welfare function more than a computee with a different behaviour profile, but also vice-versa. Moreover, a trade-off between "action-" and "time-effectiveness" must be expected, according to the delays that may be introduced by deferring action executions. Also, the environment accessibility influences the effectiveness of the profile.

Alternative definitions.

- *Focused and cautious* computee. Pursues a (top) goal at the time in a (actively) cautious style: does not perform any action of the plan for the currently addressed goal, without knowing (checking, if actively cautious) that preconditions hold (see the Actively Cautious and Focused profiles).

- *Extremely cautious* computee. Checks that all the preconditions of the actions of a (partial) plan for a (top) goal hold before committing to the execution of any of the actions. It does not waste time in partially executing a plan, without being sure that the plan is "globally executable".

- *Punctually cautious* computee. Checks preconditions according to the urgency of the actions to which they refer (see the Punctual profile).

3.4 Cautious Cycle Theory based on the behaviour part

In this section we try to create a cycle theory that implements the cautious profile of behaviour by modifying the behaviour part of the cycle theory. We examine two approaches. The first approach creates a cautious cycle theory by extending the normal cycle theory. The second approach defines the cautious cycle theory as any cycle theory that satisfies certain conditions on its behaviour part.

3.4.1 A Cautious Cycle Theory based on the Normal Cycle Theory

In this section we try to create a cycle theory that implements the cautious profile by extending the *normal cycle theory* presented in D8 [3]. In order to do that we first identify

the characteristic feature of the cautious profile and then extend the *normal cycle theory* in order to create a cycle theory that induces an operational trace which has the required feature. Finally we prove that the proposed cycle theory does induce such operational traces.

The cautious cycle theory is the *normal cycle theory* with the following two rules added to the $\mathcal{T}_{behaviour}$ part of the theory:

- $Cautious\_\mathcal{P}^T_{T1 \succ AE} : \mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, As) \leftarrow unknown\_pred(S, As)$,
  for every T and $T1, X$ such that either $T1 \neq AE$ or $T1 = AE$ and $X \neq As$, where the predicate $unknown\_pred(S, As)$ is defined appropriately using the Temporal Reasoning capability of the computee.

- $Cautious\_\mathcal{MP}^T_{T1 \succ AE} : Cautious\_\mathcal{P}^T_{T1 \succ AE} \succ \mathcal{P}^T_{AE \succ T1}$
  for every T and $T1, X$ such that $T1 \neq AE$.

The purpose of the first rule is to give lower priority over any other transition to transitions of Action Execution with input parameter a set of actions A which contains an action whose preconditions are not believed to be satisfied at the current state and time. The purpose of the second rule is to give the first rule higher priority than any other priority rule in the behaviour part of the computee's cycle theory.

Proposition 3.2 *The cautious cycle theory induces an operational trace which has the characteristic feature of the cautious profile.*

This proposition says that a computee with the *cautious cycle theory* will not choose to execute an action if it is not sure that all its preconditions hold at the current state.

Formally

$$\mathcal{T}_{cycle} \wedge T_i(S_{i-1}, X_i, S_i) \nvdash_{pr} AE(S_i, A)$$

for some action $A = \langle a[t], \_, C \rangle$, if $\exists c \in C$ for which $KB^i \nvdash_{TR} c$.

In other words an action execution is not an admissible conclusion of the $\mathcal{T}_{cycle}$ theory of the cautious computee, if the preconditions of that action are not known to be true at the time of the execution.

Summary of Proof In order to prove the proposition we will try to construct an admissible set of rules which concludes the transition $AE(S_i, A)$, assuming that $KB^i \nvdash_{TR} c$ holds, where $A = \langle a[t], \_, C \rangle$, $c \in C$. When we fail to do so, we will have proven that it is not an admissible conclusion and therefore the cycle theory satisfies the required property.

To do this we consider all different rules in the $\mathcal{T}_{basic}$ part of the cycle theory that enable action execution as the next transition. For each one of these rules we consider all possible extensions of it with priority rules and for each extension, we construct a set that attacks it and cannot be attacked back. Since there exists such a set for each one of the possible extensions, we conclude that none of them is admissible.

Since there does not exist an admissible set of rules which concludes the action execution transition when its preconditions do not hold, the transition is not admissible and therefore the proposition holds.

Proof According to the *cautious cycle theory*, the rules which decide when an action execution transition should follow are:

- $\mathcal{R}_{AE|AE}(S', As') : *AE(S', As') \leftarrow AE(S, As, S'), As' = c_{AS}(S', \tau), As' \neq \{\}$
- $\mathcal{R}_{PI|AE}(S', As) : *AE(S', As) \leftarrow PI(S, Gs, S'), As = c_{AS}(S', \tau), As \neq \{\}$
- $\mathcal{R}_{SI|AE}(S', As) : *AE(S', As) \leftarrow SI(S, Ps, S'), As = c_{AS}(S', \tau), As \neq \{\}$
- $\mathcal{R}_{AOI|AE}(S', As) : *AE(S', As) \leftarrow AOI(S, Fs, S'), As = c_{AS}(S', \tau), As \neq \{\}$

We know that these four rules are incompatible with each other because of the rule $incompatible(T(S, X), T'(S, X'))$ in $\mathcal{T}_{cycle}$. Since an admissible set has to be consistent, we cannot have more than one of these rules in an admissible set.

A case analysis follows during which we try (and fail) to construct an admissible set of rules, using each one of the preceding rules.

Let's try to construct an admissible set of rules using the first one of these rules ($\mathcal{R}_{AE|AE}(S', As')$). Let's first show that a set Δ consisting of only this rule is not admissible. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks. Let's consider the following set of rules: $\Delta' = \{\mathcal{R}_{AE|PI}(S', Gs),\ Cautious\_\mathcal{P}^T_{T1 \succ AE},\ Cautious\_\mathcal{MP}^T_{T1 \succ AE}\}$ where:

- $\mathcal{R}_{AE|PI}(S', Gs) : *PI(S', Gs) \leftarrow AE(S, As, S'), Gs = c_{GS}(S', \tau), Gs \neq \{\}$
- $Cautious\_\mathcal{P}^T_{T1 \succ AE} : \mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, As) \leftarrow unknown\_pred(S, As)$
- $Cautious\_\mathcal{MP}^T_{T1 \succ AE} : Cautious\_\mathcal{P}^T_{T1 \succ AE} \succ \mathcal{P}^T_{AE \succ T1}$

We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash AE(S', As')$, $\Delta' \vdash PI(S', Gs)$ and $incompatible(AE(S', As'), PI(S', Gs))$ holds. Since there is no rule in Δ that has higher priority than any rule in Δ', Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $\mathcal{R}_{AE|PI}(S', Gs)$ in Δ' has higher priority than the rule $\mathcal{R}_{AE|AE}(S', As')$ in Δ according to the rule $Cautious\_\mathcal{P}^T_{T1 \succ AE}$ in Δ'. But Δ does not have a rule with higher priority than any rule in Δ'. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

Let's now try to extend the set Δ by adding to it the priority rule $\mathcal{P}^{AE}_{AE \succ T} : \mathcal{R}_{AE|AE}(S, As) \succ \mathcal{R}_{AE|T}(S, X)$. Now $\Delta = \{\mathcal{R}_{AE|AE}(S', As'),\ \mathcal{P}^{AE}_{AE \succ T}\}$. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks. Consider the set $\Delta_2 \subseteq \Delta'$ where $\Delta_2 = \{Cautious\_\mathcal{P}^T_{T1 \succ AE}, Cautious\_\mathcal{MP}^T_{T1 \succ AE}\}$.

We first need to show that Δ' attacks Δ. The sets Δ and Δ_2 draw opposite conclusions since $\Delta \vdash \mathcal{R}_{AE|AE}(S, As) \succ \mathcal{R}_{AE|T}(S, X)$, $\Delta_2 \vdash \mathcal{R}_{AE|T}(S, X) \succ \mathcal{R}_{AE|AE}(S, As)$. Since there is no rule in Δ that has higher priority than any rule in Δ_2, Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $Cautious\_\mathcal{P}^T_{T1 \succ AE}$ in Δ_2 has higher priority than the rule $\mathcal{P}^{AE}_{AE \succ T}$ in Δ, according to the rule $Cautious\_\mathcal{MP}^T_{T1 \succ AE}$ in Δ_2. But Δ does not have a rule with higher priority than any rule in Δ_2. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

Note that no other possible extension exists for the rule $\mathcal{R}_{AE|AE}(S', As')$ since there are no other priority rules that we could add that would create a consistent set.

Since we failed to construct an admissible set using the rule $\mathcal{R}_{AE|AE}(S', As')$, let's try to construct an admissible set of rules that concludes $AE(S', As)$ using the rule $\mathcal{R}_{PI|AE}(S', As)$.

Let's first show that a set Δ consisting of only this rule is not admissible. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks. Let's consider the following set of rules: $\Delta' = \{\mathcal{R}_{PI|SI}(S', Ps),\ \mathcal{P}^{PI}_{SI\succ T}\}$ where:

- $\mathcal{R}_{PI|SI}(S', Ps) : *SI(S', Ps) \leftarrow PI(S, Gs, S'), Ps = c_{PS}(S', \tau), Ps \neq \{\}$
- $\mathcal{P}^{PI}_{SI\succ T} : \mathcal{R}_{PI|SI}(S, Ps) \succ \mathcal{R}_{PI|T}(S, As) \leftarrow unreliable\_pre(As)$

We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash AE(S', As)$, $\Delta' \vdash SI(S', Ps)$ and $incompatible(AE(S', As'), SI(S', Ps))$ holds. Since there is no rule in Δ that has higher priority than any rule in Δ', Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $\mathcal{R}_{PI|SI}(S', Ps)$ in Δ' has higher priority than the rule $\mathcal{R}_{PI|AE}(S', As)$ in Δ, according to the rule $\mathcal{P}^{PI}_{SI\succ T}$ in Δ'. But Δ does not have a rule with higher priority than any rule in Δ'. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

Note that we cannot extend Δ by adding the priority rule

$$\mathcal{P}^{PI}_{AE\succ T} : \mathcal{R}_{PI|AE}(S, As) \succ \mathcal{R}_{PI|T}(S, X) \leftarrow not\ unreliable\_pre(As)$$

to it, since $unreliable\_pre(As)$ holds. Note also that no possible extension exists for the rule $\mathcal{R}_{AE|AE}(S', As')$ since there are no other priority rules that we could add that would create a consistent set.

Since we failed to construct an admissible set using the rule $\mathcal{R}_{PI|AE}(S', As)$, let's try to construct an admissible set of rules that concludes $AE(S', As)$ using the rule $\mathcal{R}_{SI|AE}(S', As)$. Let's first show that a set Δ consisting of only this rule is not admissible. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks. Let's consider the following set of rules: $\Delta' = \{\mathcal{R}_{SI|PR}(S'),\ Cautious\_\mathcal{P}^{T}_{T1\succ AE},\ Cautious\_\mathcal{MP}^{T}_{T1\succ AE}\}$ where:

- $\mathcal{R}_{SI|PR}(S') : *PR(S') \leftarrow SI(S, Ps, S')$
- $Cautious\_\mathcal{P}^{T}_{T1\succ AE} : \mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, As) \leftarrow unknown\_pred(S, As)$
- $Cautious\_\mathcal{MP}^{T}_{T1\succ AE} : Cautious\_\mathcal{P}^{T}_{T1\succ AE} \succ \mathcal{P}^{T}_{AE\succ T1}$

We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash AE(S', As')$, $\Delta' \vdash PR(S')$ and $incompatible(AE(S', As'), PR(S'))$ holds. Since there is no rule in Δ that has higher priority than any rule in Δ', Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $\mathcal{R}_{SI|PR}(S')$ in Δ' has higher priority than the rule $\mathcal{R}_{SI|AE}(S', As)$ in Δ according to the rule $Cautious\_\mathcal{P}^{T}_{T1\succ AE}$ in Δ'. But Δ does not have a rule with higher priority than any rule in Δ'. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

Let's now try to extend the set Δ by adding to it the priority rule $\mathcal{P}^{SI}_{AE\succ T} : \mathcal{R}_{SI|AE}(S, As) \succ \mathcal{R}_{SI|T}(S, X)$. Now $\Delta = \{\mathcal{R}_{SI|AE}(S', As),\ \mathcal{P}^{SI}_{AE\succ T}\}$. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks. Consider the set $\Delta_2 \subseteq \Delta'$ where $\Delta_2 = \{Cautious\_\mathcal{P}^{T}_{T1\succ AE}, Cautious\_\mathcal{MP}^{T}_{T1\succ AE}\}$.

We first need to show that Δ' attacks Δ. The sets Δ and Δ_2 draw opposite conclusions since $\Delta \vdash \mathcal{R}_{SI|AE}(S, As) \succ \mathcal{R}_{SI|T}(S, X)$, $\Delta_2 \vdash \mathcal{R}_{SI|T}(S, X) \succ \mathcal{R}_{SI|AE}(S, As)$. Since there is no rule in Δ that has higher priority than any rule in Δ_2, Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $Cautious\_\mathcal{P}^{T}_{T1\succ AE}$ in Δ_2 has higher priority than the rule $\mathcal{P}^{SI}_{AE\succ T}$ in Δ, according to the rule $Cautious\_\mathcal{MP}^{T}_{T1\succ AE}$ in Δ_2.

But Δ does not have a rule with higher priority than any rule in Δ_2. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

Note that no other possible extension exists for the rule $\mathcal{R}_{SI|AE}(S', As)$ since there are no other priority rules that we could add that would create a consistent set.

Since we failed to construct an admissible set using the rule $\mathcal{R}_{SI|AE}(S', As)$, let's try to construct an admissible set of rules that concludes $AE(S', As)$ using the rule $\mathcal{R}_{AOI|AE}(S', As)$. Let's first show that a set Δ consisting of only this rule is not admissible. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks. Let's consider the following set of rules: $\Delta' = \{\mathcal{R}_{AOI|GR}(S'), Cautious\_\mathcal{P}^T_{T1\succ AE}\}$ where:

- $\mathcal{R}_{AOI|GR}(S') : *GR(S') \leftarrow AOI(S, Fs, S')$
- $Cautious\_\mathcal{P}^T_{T1\succ AE} : \mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, As) \leftarrow unknown\_pred(S, As)$

We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash AE(S', As')$, $\Delta' \vdash GR(S')$ and $incompatible(AE(S', As'), GR(S'))$ holds. Since there is no rule in Δ that has higher priority than any rule in Δ', Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $\mathcal{R}_{AOI|GR}(S')$ in Δ' has higher priority than the rule $\mathcal{R}_{AOI|AE}(S', As)$ in Δ, according to the rule $Cautious\_\mathcal{P}^T_{T1\succ AE}$ in Δ'. But Δ does not have a rule with higher priority than any rule in Δ'. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

Note that no possible extension exists for the rule $\mathcal{R}_{AOI|AE}(S', As)$ since there are no other priority rules that we could add that would create a consistent set.

Since we failed to create an admissible set of rules which concludes the transition action execution, when there are preconditions which are not known to be true at the time of the execution, we can conclude that this transition is not an admissible conclusion of the cautious cycle theory. Therefore we have proven that a computee with the cautious cycle theory satisfies the required property.

3.4.2 Cautious Cycle Theories

In the previous section we created a cycle theory that implements the cautious profile by adding two priority rules to the $\mathcal{T}_{behaviour}$ part of the *normal cycle theory*. In this section we prove that we can achieve the cautious profile be adding the two priority rules to *any* cycle theory. In order to do that we redefine the definition and characteristic feature of the cautious profile for reasons explained bellow. We then give a definition for what a *cautious cycle theory* is and prove that any such theory induces an operational trace which has the required feature.

Informal Definition. A *cautious* profile of behaviour requires a computee to execute actions in its plan only if it believes that the preconditions of these actions are currently true or if the only transitions currently enabled are action execution transitions for which the preconditions are not known to hold.

Characteristic Feature of the Cautious Profile(2). Given any transition $T_i(S_{i-1}, X_i, S_i, \tau_i)$ in the operational trace of the computee such that $executed(a[t], \tau_i) \in KB^i_0$ for some action $A = \langle a[t], \_, C\rangle$, where $S_i = \langle KB^i, Goals^i, Plan^i, TCS^i\rangle$, then

- either $KB^{i-1} \vDash_{TR} C \wedge \Sigma[S_{i-1}] \wedge t = \tau_{i-1}$

- or the only transitions possible from state S_{i-1} are of the form $AE(S_{i-1}, A)$, where $A = \langle a[t], \_, C\rangle$, and $\exists c \in C$ such that $KB^{i-1} \nvDash_{TR} c$.

In other words the computee does not execute an action if it does not know that its preconditions hold, unless it has no other choice of transition. In the previous section we used rules from the $\mathcal{T}_{basic}$ part of the *normal cycle theory* which enabled other transitions whenever an action execution transition was enabled, in order to prove that proposition 3.2 holds. Since now we do not know if there are such rules in the cycle theory, we allow the computee to execute actions with uncertain preconditions if there is no other transition enabled. Alternatively we could leave the characteristic feature unchanged and change the cycle theory definition by adding a condition that ensures that whenever an action execution whose preconditions do not hold is enabled, another transition (which is either not an action execution, or it is an action execution whose preconditions hold) is also enabled.

A cautious cycle theory is *any* cycle theory where

- The following two rules are part of the $\mathcal{T}_{behaviour}$ part of the theory:
  - $Cautious\_\mathcal{P}^T_{T1 \succ AE} : \mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, As) \leftarrow unknown\_pred(S, As)$, for every T and $T1, X$ such that either $T1 \neq AE$ or $T1 = AE$ and $X \neq As$, where the predicate $unknown\_pred(S, As)$ is defined appropriately using the Temporal Reasoning capability of the computee.
  - $Cautious\_\mathcal{MP}^T_{T1 \succ AE} : Cautious\_\mathcal{P}^T_{T1 \succ AE} \succ \mathcal{P}^T_{AE \succ T1}$, for every T and $T1, X$ such that $T1 \neq AE$.
- There is no rule which could enable $\mathcal{P}^T_{AE \succ T1} \succ Cautious\_\mathcal{P}^T_{T1 \succ AE}$ in the $\mathcal{T}_{behaviour}$ part of the cycle theory.

Proposition 3.3 *A cautious cycle theory induces an operational trace which has the characteristic feature of the cautious profile.*

Proof In order to prove the proposition we will try to construct an admissible set of rules which concludes the transition $AE(S_i, A)$, assuming that $KB^i \nvDash_{TR} c$ holds, where $A = \langle a[t], \_, C\rangle$, $c \in C$ and that there exists a possible transition $T'(S_i, X)$, where either $T' \neq AE$ or $T' = AE$, $X = \langle a'[t], \_, C'\rangle$ and $KB^i \vDash_{TR} C'$. When we fail to do so, we will have proven that it is not an admissible conclusion and therefore the cycle theory satisfies the required property.

The set must contain a base rule of the form $\mathcal{R}_{T|AE}(S, X)$ that enables an action execution as the next transition. Let's assume that such a rule exists and is enabled at the current state. Let's first show that a set Δ consisting of only this rule is not admissible. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks.

Let's now try and construct a set Δ' that attacks Δ. Based on our assumption, there exists a possible transition $T'(S_i, X')$, which means that there exists a rule $\mathcal{R}_{T|T'}(S, X)$ which is currently enabled. Let's add this rule to the set Δ' along with the two rules that were added to the cycle theory in order to achieve the cautious profile: $\Delta' = \{\mathcal{R}_{T|T'}(S, X'), Cautious\_\mathcal{P}^T_{T1 \succ AE}, Cautious\_\mathcal{MP}^T_{T1 \succ AE}\}$.

We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash AE(S, X)$, $\Delta' \vdash T'(S, X')$, where $AE \neq T'$ or $X \neq X'$, and $incompatible(AE(S, X), T'(S, X'))$ holds. Since there is no rule in Δ that has higher priority than any rule in Δ', Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $\mathcal{R}_{T|T'}(S, X')$in Δ' has higher priority than the rule $\mathcal{R}_{T|AE}(S, X)$ in Δ according to the rule $Cautious\_\mathcal{P}^T_{T1\succ AE}$ in Δ'. But Δ does not have a rule with higher priority than any rule in Δ'. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

In order to extend the set Δ let's assume that there exists a priority rule in the $\mathcal{T}_{behaviour}$ part of the cycle theory that gives higher priority to the rule $\mathcal{R}_{T|AE}(S, X)$ over any other rule that is currently enabled. The constructed set is $\Delta = \{\mathcal{R}_{T|AE}(S, X), \mathcal{P}^T_{AE\succ T1}\}$. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks. Consider the set $\Delta_2 \subseteq \Delta'$ where $\Delta_2 = \{Cautious\_\mathcal{P}^T_{T1\succ AE}, Cautious\_\mathcal{MP}^T_{T1\succ AE}\}$.

We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash \mathcal{R}_{T|AE}(S, X) \succ \mathcal{R}_{T|T1}(S, X)$, $\Delta_2 \vdash \mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, X)$. Since there is no rule in Δ that has higher priority than any rule in Δ_2, Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $Cautious\_\mathcal{P}^T_{T1\succ AE}$ in Δ_2 has higher priority than the rule $\mathcal{P}^T_{AE\succ T1}$ in Δ, according to the rule $Cautious\_\mathcal{MP}^T_{T1\succ AE}$ in Δ_2. But Δ does not have a rule with higher priority than any rule in Δ_2. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

Note that there is no rule that we could add in the set Δ that would make it admissible, since according to the rule $Cautious\_\mathcal{MP}^T_{T1\succ AE}$ the rule $Cautious\_\mathcal{P}^T_{T1\succ AE}$ has higher priority over any other priority rule and according to the cycle theory definition there is no rule in $\mathcal{T}_{behaviour}$ that gives higher priority to a priority rule over the rule $Cautious\_\mathcal{P}^T_{T1\succ AE}$.

Since we failed to create an admissible set of rules which concludes the transition action execution, when there are preconditions which are not known to be true at the time of the execution and when there are other transitions enabled, we can conclude that this transition is not an admissible conclusion of the cautious cycle theory. Therefore we have proven that a computee with a cautious cycle theory satisfies the required property.

4 Actively Cautious Profile

In this section we study the actively cautious profile of behaviour. In section 4.2 we identify the characteristic feature of the actively cautious profile. In section 4.3 we define what an *actively cautious cycle theory* is and prove that any such cycle theory induces an operational trace which has the required feature. Finally in section 4.4 we define and prove comparison properties for the actively cautious profile.

4.1 Informal Definition

An *actively cautious* profile of behaviour requires a computee to execute actions in its plan only if it believes that the preconditions of these actions are currently true (as in the cautious case). In addition, the actively cautious computee executes sensing actions in order to actively obtain the value of unknown preconditions.

4.2 Formal Definition

Characteristic Feature of the Actively Cautious Profile. Given any transition $T_i(S_{i-1}, X_i, S_i, \tau_i)$ in the operational trace of the computee where S_i =

$\langle KB^i, Goals^i, Plan^i, TCS^i \rangle$ and T_i is an action execution transition for a set of actions As, then

- For every action $A_j \in As$ where $A_j = \langle a_j[t], \_, C_j \rangle$ the following condition holds:

$$KB_{TR}^{i-1} \vDash_{TR} C_j \wedge \Sigma[S_{i-1}] \wedge t = \tau_{i-1}$$

- and, if there exists an action $A' \in c_{AS}(S_i, \tau_i)$ where $A' = \langle a'[t], \_, C' \rangle$ and there exists a maximal non-empty set of preconditions $C'' \subseteq C'$ such that for each $c \in C''$:

  - $KB_{TR}^{i-1} \nvDash_{TR} c \wedge \Sigma[S_{i-1}] \wedge t = \tau_{i-1}$
  - $KB_{TR}^{i-1} \nvDash_{TR} \neg c \wedge \Sigma[S_{i-1}] \wedge t = \tau_{i-1}$

  then all actions in the set As are necessarily sensing actions.

In other words, whenever a computee executes a set of actions, two conditions hold. Firstly, all the preconditions of those actions are known to hold at the time of the execution. Secondly, the actions executed are all sensing actions, if there exist actions in the set of selected actions for the current state whose preconditions are unknown. Notice that this condition prevents the computee from executing non-sensing actions whose preconditions are known to hold, when there exist other actions in the set of selected actions with unknown preconditions. This does not necessarily mean that the computee will execute sensing actions for all the unknown preconditions of all the actions in its plan before it starts executing the actions. Its actual behaviour depends on the precise definition of the selection function and the extend to which this selects a maximal set of the actions in the current plan. For example, if the selection function takes into account the time ordering of actions and does not select together actions which are totally ordered at different times then if there exist two actions, A_1 and A_2, whose preconditions are not known to hold, and action A_1 must be executed before action A_2, then the computee will not sense for the preconditions of the action A_2 until A_1 is executed.

We study a special case where we assume that only one action is executed at a time[5]. Note that this sets a constraint on the input parameters of the AE transition and not on the selection function. The results and proofs that follow are based on this assumption.

4.3 Actively Cautious Cycle Theories

An actively cautious cycle theory is *any* cycle theory where

- The following rules are part of the $\mathcal{T}_{behaviour}$ part of the theory:

  - $Actively\_Cautious\_\mathcal{P}^{PI}_{SI \succ T} : \mathcal{R}_{PI|SI}(S, Ps) \succ \mathcal{R}_{PI|T}(S, X)$
    for all transitions $T \neq SI$
    This rule gives higher priority to a SI transition over any other transition, when the last transition was PI.

[5]This might be too strong an assumption. Alternatively we could say that if a set of actions SAs is executed by an AE transition, then either all the actions in SAs are sensing actions, or there is no sensing action in the set SAs.

- $Actively\_Cautious\_\mathcal{MP}^{PI}_{SI\succ T} : Actively\_Cautious\_\mathcal{P}^{PI}_{SI\succ T} \succ \mathcal{P}^{PI}_{T\succ SI}$,
  for every T such that $T \neq SI$.
  This rule gives higher priority to the previous rule, over any priority rule that gives priority to a transition that is not SI, when the last transition was PI.
- $Actively\_Cautious\_\mathcal{P}^{T}_{AE\succ AE} : \mathcal{R}_{T|AE}(S, A) \succ \mathcal{R}_{T|AE}(S, X) \leftarrow A = \langle sense\_precondition(c[t]), \_, \_\rangle, \exists A' \in c_{AS}(S, \tau), A' = \langle a[t], \_, C\rangle, c \in C, unknown(c), now(\tau)$.
  for every T, X such that X is not a sensing action, where the predicate $unknown(c)$ is defined appropriately using the Temporal Reasoning capability of the computee.
  This rule gives higher priority to an AE transition where the action is a sensing action, over any other AE transition, when there exist actions in the set of selected actions with unknown preconditions.
- $Actively\_Cautious\_\mathcal{MP}^{T}_{AE\succ AE} : Actively\_Cautious\_\mathcal{P}^{T}_{AE\succ AE} \succ \mathcal{P}^{T}_{AE\succ AE}$
  for every T and $\mathcal{P}$ such that $\mathcal{P} \neq Actively\_Cautious\_\mathcal{P}$.
  This rule gives higher priority to the previous rule over any rule that gives priority to an AE transition, over some other AE transition.
- $Cautious\_\mathcal{P}^{T}_{T1\succ AE} : \mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, As) \leftarrow unknown\_pred(S, As)$,
  for every T and $T1, X$ such that either $T1 \neq AE$ or $T1 = AE$ and $X \neq As$, where the predicate $unknown\_pred(S, As)$ is defined appropriately using the Temporal Reasoning capability of the computee.
  This rule gives lower priority, over any other transition, to AE transitions with input parameter a set of actions which contains an action whose preconditions are not known to hold at the current state and time.
- $Cautious\_\mathcal{MP}^{T}_{T1\succ AE} : Cautious\_\mathcal{P}^{T}_{T1\succ AE} \succ \mathcal{P}^{T}_{AE\succ T1}$,
  for every T and $T1, X$ such that $T1 \neq AE$.
  This rule gives higher priority to the previous rule over any other priority rule that gives higher priority to an AE transition.

- The following rule is part of the $\mathcal{T}_{basic}$ part of the theory

  $$\mathcal{R}_{PI|SI}(S', Ps) : *SI(S', Ps) \leftarrow PI(S, Gs, S', \tau), Ps = c_{PS}(S', \tau'), Ps \neq \{\}, now(\tau')$$

  This rule enables a SI transition to follow a PI transition. We assume that the set of preconditions of each of the sensing actions added to the plan by this rule, is empty.

- There is no rule which could enable $\mathcal{P}^{PI}_{T\succ SI} \succ Actively\_Cautious\_\mathcal{P}^{PI}_{SI\succ T}$ in the $\mathcal{T}_{behaviour}$ part of the cycle theory.
- There is no rule which could enable $\mathcal{P}^{T}_{AE\succ T1} \succ Cautious\_\mathcal{P}^{T}_{T1\succ AE}$ in the $\mathcal{T}_{behaviour}$ part of the cycle theory.
- There is no rule that could enable $\mathcal{P}^{T}_{AE\succ AE} \succ Actively\_Cautious\_\mathcal{P}^{T}_{AE\succ AE}$, where $\mathcal{P} \neq Actively\_Cautious\_\mathcal{P}$.

Proposition 4.1 *An actively cautious cycle theory induces an operational trace which has the characteristic feature of the actively cautious profile.*

Summary of Proof We need to prove that for every action execution transition in the operational trace of the computee, two conditions hold:

1. All the preconditions of the action are known to hold at the time of the execution.
2. If there exist actions in the set of selected actions whose preconditions are not known to hold, then the action executed is a sensing action.

In order to prove that these conditions hold we will first prove two auxiliary propositions:

Proposition 4.2 *Whenever a PI transition takes place in the operational trace induced by an actively cautious cycle theory, if there exist actions in the plan whose preconditions are unknown, then the next transition is always a SI transition.*

Proposition 4.3 *Whenever there exist actions in the set of selected actions, in the operational trace induced by the actively cautious cycle theory, whose preconditions are not known to hold, and the last transition was not a PI transition, then there exists at least one sensing action in the set of selected actions.*

After we prove that these propositions hold, we will prove the two conditions stated above. In order to prove the first condition we will try to construct an admissible set of rules which concludes an action execution transition, for some action whose preconditions are not known to hold. When we fail to do so, we will have proved that the cycle theory satisfies the first condition.

In order to prove the second condition we will prove that there does not exist an admissible set of rules which concludes an action execution which is not a sensing action, when there exist actions with unknown preconditions in the set of selected actions.

Proof

- *Whenever a PI transition takes place, if there exist actions in the plan whose preconditions are unknown, then the next transition is always a SI transition.*

  Proof. In order to prove this we will try to construct an admissible set of rules which concludes a transition T following PI, where $T \neq SI$. When we fail to do so we will have proven that SI is the only admissible conclusion and therefore SI is always executed after PI when there exist actions with unknown preconditions.

  The set must contain a base rule of the form $\mathcal{R}_{PI|T}(S, X)$ where $T \neq SI$, that enables a transition after transition PI that is not a sensing introduction transition. Let's first show that a set Δ consisting of only this rule is not admissible. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks.

  Let's now try and construct a set Δ' that attacks Δ. Based on the definition of an actively cautious cycle theory, there exists a rule $\mathcal{R}_{PI|SI}(S, Ps)$ in the basic part of the theory. Let's add this rule to the set Δ' along with two of the priority rules from the definition of the actively cautious profile: $\Delta' = \mathcal{R}_{PI|SI}(S', Ps), Actively\_Cautious\_\mathcal{P}^{PI}_{SI \succ T}, Actively\_Cautious\_\mathcal{MP}^{PI}_{SI \succ T}\}$.

  We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash T(S, X)$, $\Delta' \vdash SI(S, Ps)$, where $T \neq SI$ and $incompatible(T(S, X), SI(S, Ps))$ holds. Since there is no rule in Δ that has higher priority than any rule in Δ', Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $\mathcal{R}_{PI|SI}(S', Ps)$ in Δ' has higher priority than the rule $\mathcal{R}_{PI|T}(S, X)$ in Δ according to the rule $Actively\_Cautious\_\mathcal{P}^{PI}_{SI\succ T}$ in Δ'. But Δ does not have a rule with higher priority than any rule in Δ'. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

In order to extend the set Δ let's assume that there exists a priority rule in the $\mathcal{T}_{behaviour}$ part of the cycle theory that gives higher priority to the rule $\mathcal{R}_{PI|T}(S, X)$ over any other rule that is currently enabled. The constructed set is $\Delta = \{\mathcal{R}_{PI|T}(S, X), \mathcal{P}^{PI}_{T\succ T1}\}$. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks. Consider the set $\Delta_2 \subseteq \Delta'$ where $\Delta_2 = \{Actively\_Cautious\_\mathcal{P}^{PI}_{SI\succ T}, Actively\_Cautious\_\mathcal{MP}^{PI}_{SI\succ T}\}$.

We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash \mathcal{R}_{PI|T}(S, X) \succ \mathcal{R}_{PI|SI}(S, Ps)$, $\Delta_2 \vdash \mathcal{R}_{PI|SI}(S, Ps) > \mathcal{R}_{PI|T}(S, X)$. Since there is no rule in Δ that has higher priority than any rule in Δ_2, Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $Actively\_Cautious\_\mathcal{P}^{PI}_{SI\succ T}$ in Δ_2 has higher priority than the rule $\mathcal{P}^{PI}_{T\succ T1}$ in Δ, according to the rule $Actively\_Cautious\_\mathcal{MP}^{PI}_{SI\succ T}$ in Δ_2. But Δ does not have a rule with higher priority than any rule in Δ_2. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

Note that there is no rule that we could add in the set Δ that would make it admissible, since according to the rule $Actively\_Cautious\_\mathcal{MP}^{PI}_{SI\succ T}$ the rule $Actively\_Cautious\_\mathcal{P}^{PI}_{SI\succ T}$ has higher priority over any other priority rule of the form $\mathcal{P}^{PI}_{T\succ SI}$. Also, according to the cycle theory definition there is no rule in $\mathcal{T}_{behaviour}$ that gives higher priority to any priority rule $\mathcal{P}^{PI}_{T\succ SI}$ over the rule $Actively\_Cautious\_\mathcal{P}^{PI}_{SI\succ T}$.

Since we failed to create an admissible set of rules which concludes that a transition T which is not a sensing introduction transition follows a PI transition, we can conclude that this transition is not an admissible conclusion of the actively cautious cycle theory. Therefore we have proven that a computee with an actively cautious cycle theory, always executes a SI transition after a PI, whenever actions with unknown precondition exist in the current plan.

- *Whenever there exist actions in the set of selected actions whose preconditions are not known to hold, and the last transition was not a PI transition, then there exists at least one sensing action in the set of selected actions.*

  Proof. Let's assume that there exists an action A in the set of selected actions for the current state whose preconditions are not known to hold, and that the previous transition was not a PI transition. We will prove that there exists at least one sensing action in the set of selected actions.

  We have already proven that after a PI transition takes place, the next transition is SI, which introduces sensing actions for all the unknown preconditions of all the actions in the plan. Since action A is in the set of selected actions, it is also in the current plan. We have assumed that the last transition was not PI, so we know that when PI was executed for the current plan, SI added sensing actions in the current plan for all the unknown preconditions of all the actions in the plan. Since the preconditions of action A are still unknown, then the sensing action is still in the plan and in the set of selected

actions.

- *For every action execution transition in the operational trace of the computee, all the preconditions of the action are known to hold at the time of the execution.*

  The proof is the same as in the cautious profile.

- *For every action execution transition in the operational trace of the computee, if there exist actions in the set of selected actions whose preconditions are not known to hold, then the action executed is a sensing action.*

  Proof. Let T_i be a transition in the operational trace of the computee where $T_i(S_{i-1}, X_i, S_i, \tau_i)$, $S_i = \langle KB^i, Goals^i, Plan^i, TCS^i \rangle$ and T_i is an action execution transition for some action $A = \langle a[t], \_, C \rangle$. Let's assume that there exists an action $A' \in c_{AS}(S_{i-1}, \tau_{i-1})$ where $A' = \langle a'[t], \_, C' \rangle$ and there exists a maximal non-empty set of preconditions $C'' \subseteq C'$ such that for each $c \in C''$:

  - $KB^{i-1} \nvdash_{TR} c \wedge \Sigma[S_{i-1}] \wedge t = \tau_{i-1}$
  - $KB^{i-1} \nvdash_{TR} \neg c \wedge \Sigma[S_{i-1}] \wedge t = \tau_{i-1}$

  In order to prove the proposition we will try to construct an admissible set of rules which concludes that action A is not a sensing action. When we fail to do so, we will have proven that it is not an admissible conclusion and therefore the cycle theory satisfies the required property.

  The set must contain a base rule of the form $\mathcal{R}_{T|AE}(S, X)$ that enables an action execution as the next transition, for some action X which is not a sensing action. Since we already know that the transition T_i is an action execution transition, we know that such a rule exists and is enabled at the current state. Let's first show that a set Δ consisting of only this rule is not admissible. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks.

  Let's now try and construct a set Δ' that attacks Δ. We have already proven that whenever a PI transition takes place, if there exist actions in the plan whose preconditions are unknown, then the next transition is always a SI transition. Since transition T_i is an AE transition and there exists an action in the set of selected actions whose preconditions are not known to hold, we know that transition T_{i-1} was not a PI transition. We have also proven that when the last transition is not a PI transition and there exist actions in the set of selected actions whose preconditions are not known to hold, then there exists at least one sensing action in the set of selected actions. Also, as stated above, we already know that a rule $\mathcal{R}_{T|AE}$ exists and is enabled at the current state and time. Therefore we can assume that the sensing action that exists in the set of selected actions is enabled by this rule. Let's add this rule to the set Δ' along with two of the rules that were added to the behaviour part of the cycle theory in order to achieve the actively cautious profile: $\Delta' = \{\mathcal{R}_{T|AE}(S, X'), Actively\_Cautious\_\mathcal{P}^T_{AE \succ AE}, Actively\_Cautious\_\mathcal{MP}^T_{AE \succ AE}\}$.

  We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash AE(S, X)$, $\Delta' \vdash AE(S, X')$, where $X \neq X'$, and $incompatible(AE(S, X), AE(S, X'))$ holds. Since there is no rule in Δ that has higher priority than any rule in Δ', Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $\mathcal{R}_{T|AE}(S, X')$ in Δ' has higher priority than the rule $\mathcal{R}_{T|AE}(S, X)$ in Δ according to the rule $Actively\_Cautious\_\mathcal{P}^T_{AE\succ AE}$ in Δ'. But Δ does not have a rule with higher priority than any rule in Δ'. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

In order to extend the set Δ let's assume that there exists a priority rule in the $\mathcal{T}_{behaviour}$ part of the cycle theory that gives higher priority to the rule $\mathcal{R}_{T|AE}(S, X)$ over any other rule that is currently enabled. The constructed set is $\Delta = \{\mathcal{R}_{T|AE}(S, X), \mathcal{P}^T_{AE\succ T1}\}$. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks. Consider the set $\Delta_2 \subseteq \Delta'$ where $\Delta_2 = \{Actively\_Cautious\_\mathcal{P}^T_{AE\succ AE}, Actively\_Cautious\_\mathcal{MP}^T_{AE\succ AE}\}$.

We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash \mathcal{R}_{T|AE}(S, X) \succ \mathcal{R}_{T|AE}(S, X')$, $\Delta_2 \vdash \mathcal{R}_{T|AE}(S, X') \succ \mathcal{R}_{T|AE}(S, X)$ and $X \neq X'$. Since there is no rule in Δ that has higher priority than any rule in Δ_2, Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $Actively\_Cautious\_\mathcal{P}^T_{AE\succ AE}$ in Δ_2 has higher priority than the rule $\mathcal{P}^T_{AE\succ T1}$ in Δ, according to the rule $Actively\_Cautious\_\mathcal{MP}^T_{AE\succ AE}$ in Δ_2. But Δ does not have a rule with higher priority than any rule in Δ_2. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

Note that there is no rule that we could add in the set Δ that would make it admissible, since according to the rule $Actively\_Cautious\_\mathcal{MP}^T_{AE\succ AE}$ the rule $Actively\_Cautious\_\mathcal{P}^T_{AE\succ AE}$ has higher priority over any other priority rule that gives priority to an action execution transition over some other execution transition. Also, according to the cycle theory definition there is no rule in $\mathcal{T}_{behaviour}$ that gives higher priority to any priority rule $\mathcal{P}^T_{AE\succ AE}$ over the rule $Actively\_Cautious\_\mathcal{P}^T_{AE\succ AE}$.

Since we failed to create an admissible set of rules which concludes that action A is not a sensing action, we can conclude that this transition is not an admissible conclusion of the theory, whenever there exist actions in the set of selected actions whose preconditions are unknown.

We have proven that both conditions hold for every computee with an actively cautious cycle theory. Therefore we have proven that a computee with an actively cautious cycle theory satisfies the required property.

4.4 Properties of the Actively Cautious Profile

We defined the *actively cautious cycle theory* as any cycle theory with certain rules added and where certain conditions hold. In order to find the properties of the actively cautious profile we will compare a computee with an actively cautious cycle theory with a computee with a modified version of the same cycle theory where the extra rules which were added to achieve the actively cautious behaviour are removed. We will call such a theory an *underlying cycle theory.*

One first attempt to write a proposition regarding the properties of the actively cautious computee compared to a computee with the underlying theory is:

A computee with an *actively cautious cycle theory* is as good as a computee with an *underlying cycle theory* with the same knowledge base KB, in the sense that whenever the latter

succeeds in achieving a goal, so does the former one, in environments where:

- sensing actions always succeed in finding out the values of all the preconditions that were sensed for.
- during the delay in executing an action, due to sensing for its preconditions, there is no change of the truth value of the preconditions via some unknown factor in the environment.
- time is not critical. That is, the delay in action execution due to sensing does not result in timing-out of the action.

Before we prove the above proposition, we need to study the last condition regarding time criticality. Since we don't want to completely eliminate time as a factor, we should try and calculate how much extra time the actively cautious agent needs because of sensing. According to the definition of an actively cautious cycle theory, the actively cautious agent chooses to execute a sensing action over any other action execution transition, whenever there exist actions in the set of selected actions whose preconditions are not known to hold. In the worst case it will have to sense for all the preconditions of all the actions in the current plan before it executes the first one of these actions. Also in the worst case it will take one sensing transition for each one of these preconditions. On top of that the actively cautious agent will also execute a Sensing Introduction transition right after executing a Plan Introduction transition, whereas the computee with the underlying cycle theory might not take this step. Therefore the maximum extra transition steps that an actively cautious agent will need before executing a specific action, compared to the computee with the underlying cycle theory, will be equal to the sum of all preconditions of all the actions in the current plan plus one for the Sensing Introduction step. Hence we can modify the previous proposition as follows.

Proposition 4.4 *A computee with an actively cautious cycle theory is as good as a computee with the underlying cycle theory with the same knowledge base KB, in the sense that whenever the latter succeeds in achieving a goal, so does the former one, in environments where:*

- *sensing actions always succeed in finding out the values of all the preconditions that were sensed for.*
- *during the delay in executing an action, due to sensing for its preconditions, there is no change of the truth value of the preconditions via some unknown factor in the environment.*
- *time is not very critical. Each action A_j in a plan $Plan$, where $A_j = \langle a_j[t], \_, C_j \rangle$, has at least time T_{A_j} before it times out, where $T_{A_j} = 1 + \Sigma_{j=0}^{|Plan|} |C_j|$, where $|Plan|$ denotes the number of actions in $Plan$ and $|C_j|$ denotes the number of precondition fluents in the set C_j.*

Proof We want to prove that if there exists an operational trace $T_0, ..., T_\kappa$ induced by the underlying cycle theory such that $S_\kappa \vDash G$ for some goal G, then there exists an operational trace $T'_0, ..., T'_n$ induced by the actively cautious cycle theory such that $S'_n \vDash G$.

By comparing the underlying cycle theory with the actively cautious cycle theory we can conclude that the differences between the operational traces induced by the two cycle theories, are the following. The operational trace induced by the actively cautious cycle theory might include some extra sensing steps, it will not include action execution transitions for actions whose preconditions are unknown and an SI transition will always follow a PI transition.

We prove the proposition by showing that every action that is executed successfully in the operational trace of the computee with the underlying cycle theory is also executed successfully in the operational trace induced by the actively cautious computee.

Let A be an action that is executed successfully in the operational trace of the computee with the underlying cycle theory. The fact that A is executed successfully means that the preconditions of A were true at the time of execution. According to the differences in the behaviour of the two computees, the only difference in the actions included in their Plan, is that the plan of the actively cautious computee may include some extra sensing actions, but all the other actions in the plan will be the same. Hence action A is also in the plan of the actively cautious computee. Even though the action might be executed at a later time in the operational trace of the actively cautious computee, compared to the operational trace of the computee with the underlying cycle theory, the delay does not affect the value of the preconditions of A, as stated in the proposition, and the preconditions of action A still hold. The actively cautious computee either knows that the preconditions of A hold or it senses the environment to find out their value. Since sensing always succeeds the computee eventually finds out that the preconditions hold. Note that the delay due to sensing does not affect the preconditions value. Also based on our assumptions of the environment, action A does not time out during this time and the actively cautious computee executes it successfully.

This means that if an action is executed successfully in the operational trace of the computee with the underlying cycle theory then that action also executes successfully in the operational trace induced by the actively cautious computee. Hence when a goal is achieved in the operational trace induced by the underlying cycle theory it is also achieved in the operational trace induced by the actively cautious cycle theory, therefore proposition 4.4 holds.

Based on our study on how much extra time the actively cautious computee needs in order to execute an action and also based on the above proof we can conclude the following.

Corollary 4.1 *If a computee with an underlying cycle theory successfully executes an action A_j which belongs to a plan Plan, where $A_j = \langle a_j[t], \_, C_j \rangle$, then the actively cautious computee will successfully execute the same action after at most $1 + \Sigma_{j=0}^{|Plan|}|C_j|$ number of transition steps, where $|Plan|$ denotes the number of actions in Plan and $|C_j|$ denotes the number of precondition fluents in the set C_j.*

Since a goal is achieved after the last action in the plan for that goal succeeds, and since we proved that proposition 4.4 holds we can conclude the following.

Corollary 4.2 *If a computee with an underlying cycle theory achieves a goal G and $A_\kappa \in Plan$ is the last action in the plan for goal G that is executed, where $A_j = \langle a_j[t], \_, C_j \rangle$, then the actively cautious computee will achieve the same goal after at most $1 + \Sigma_{j=0}^{|Plan|}|C_j|$ number of transition steps, where $|Plan|$ denotes the number of actions in Plan and $|C_j|$ denotes the number of precondition fluents in the set C_j.*

We can also show that in certain cases a computee with an actively cautious cycle theory is better than a computee with an underlying cycle theory in that the first succeeds in achieving a goal while the second fails to achieve the same goal.

Example 3 *We can easily create a scenario where the actively cautious computee succeeds but the computee with the underlying cycle theory fails.Suppose that the two computees have exactly*

the same knowledge base and that there exists a set of actions $A_1, \ldots, A_n$ in the set of selected actions whose preconditions are unknown. Also assume that actions $A_1, \ldots, A_{n-1}$ belong to plans for a set of goals Gs whereas action A_n belongs to a plan for a goal $G' \notin Gs$.

According to the definition of the actively cautious computee, it will execute sensing actions since there exist actions in the set of selected actions with unknown preconditions. After executing sensing the actively cautious computee finds out that the preconditions of the actions $A_1, \ldots, A_{n-1}$ do not hold whereas the preconditions of action A_n hold. Therefore the actively cautious computee does not execute any of the actions whose preconditions do not hold and executes next action A_n. The execution succeeds and goal G' is achieved.

A possible operational trace for the computee with the underlying cycle theory in the above scenario is to begin to, unsuccessfully, execute actions $A_1, \ldots, A_{n-1}$, since it does not know that their preconditions do not hold. While it is executing these actions, action A_n times out. Consequently the computee with the underlying cycle theory fails to achieve goal G'.

Therefore we conclude that in certain scenarios a computee with an actively cautious cycle theory succeeds where a computee with an underlying cycle theory fails.

It's important to comment here that a computee with the actively cautious profile would be more efficient if its definition was extended so that once it recognized that an action will not succeed because its preconditions do not hold, it would execute a plan revision transition in order to find a new plan to execute the goal which would not include these actions. This shows the need for a new plan revision transition which would take as input the set of actions whose preconditions do not hold, so that these actions will not be included in the revised plan.

5 Punctual profile

The computees adopting this profile are committed to promptly devising and executing their plans, so as to avoid that time flux can make them "obsolete", and hence unachievable, since their actions can not be executed and their goals can not be achieved within the expected deadlines.

5.1 Informal statement of the property.

The computee selects urgent goals and urgent actions for planning and execution.

The above informal statement can be further refined in two possible interpretations, which seem worth being investigated as behaviour profiles (in the following we will use the term item for both goals and actions):

1. whenever planning or executing actions select the items that are *more urgent* than the others (*simple punctual computee*),

2. if there are *very urgent* items, i.e. items close to their deadlines, prefer Planning (PI) or/and Action Execution (AE) (*punctual computee*), possibly

   (a) with a given preference among the two, and

   (b) adopting the behaviour in 1 to determine the suitable items to which to apply the transition.

The first profile can be expressed by means of a basic cycle theory by suitably constraining the items selected by the action and goal (core) selection functions, while the second one requires a behaviour cycle theory expressing the preference for PI or AE over all the other transitions, and a preference between the two transitions themselves (if one wants to avoid conflicts generated by the incompatibility relation of the cycle theory, which, although admissible in a credulous cycle theory, would make it non-deterministic).

5.2 Formal definition

Definition of the notion of *more urgent* items. We first define when an item is more urgent than others. This notion will be used to define the simple punctual computee. Given a temporal constraint store TCS, we define the deadline d of an item $i[t]$ as the minimum time point in which it becomes timed out:

$$d = \begin{cases} \infty & \text{if} \quad \forall m \ \models_{\mathcal{R}} TCS \cup \{t >= m\} \\ \\ min\{m \mid \not\models_{\mathcal{R}} TCS \cup \{t >= m\} \ \} & \text{otherwise} \end{cases}$$

where, given S the current state of the computee, $\models_{\mathcal{R}} TCS$ stands for a *total* $\Sigma(S)$*-valuation* σ exists such that $\sigma \models_{\mathcal{R}} TCS$ and $\not\models_{\mathcal{R}} TCS$ stands for a *total* $\Sigma(S)$*-valuation* σ such that $\sigma \models_{\mathcal{R}} TCS$ does not exist (see Deliverable D8, Section 7.4 for further details).

Let *now* be the current time, $i[t_i]$ and $j[t_j]$ two items whose deadlines are d_i and d_j, respectively. The item $i[t_i]$ is *more urgent* than $j[t_j]$ (with respect to TCS), written $i[t_1] \rhd j[t_j]$, iff

$$now \leq d_i < d_j$$

i.e. both the items are not currently timed out and the deadline of $i[t]$ is less (closer) than the deadline of $j[t']$. The relation $\rhd$ is *anti-reflexive* and *transitive* (trivially $d < d$ does not hold, and from $now \leq d_1 < d_2$ ($i[t_1] \rhd i[t_2]$) and $now \leq d_2 < d_3$ ($i[t_2] \rhd i[t_3]$) it follows $now \leq d_2 < d_3$ ($i[t_1] \rhd i[t_3]$) Informally speaking, $\rhd$ induces an irreflexive chain structure over the set of items.

It is easy to observe that two items are not comparable with respect to $\rhd$ iff they have the same deadline, or at least one of them is already timed out. (Trivially, the *if* part by definition, and the *only if* part by supposing that $i[t_1]$ and $j[t_2]$ have different deadlines, and they are bigger or equal than *now*, but then $now \leq d_1 < d_2$, i.e. $i[t_1] \rhd j[t_2]$, or viceversa $now \leq d_2 < d_1$).

Given a set of items X, the time parameter *now*, the relation $more\_urgent\_items(X, Y, now)$ holds iff Y contains the maximal elements of X, with respect to $\rhd$, i.e.

$$\forall\ i[t] \in X.\ i[t] \in Y \ \Leftrightarrow \nexists\ j[t'] \in X.\ j[t'] \rhd i[t],$$

Any element in Y is closer to its deadline than any element in X. By using the $more\_urgent\_items/3$ relation to select the actions to execute in an AE transition or the goals to plan for during a PI transition induces an ordering which is "safe" with respect to TCS, but it may unnecessarily restrict the number of executed actions or goals for which a plan is devised. This could even cause some actions or goals to unnecessarily become timed out, while they could have been executed all together in one single transition. In order to overcome this problem it is possible to enlarge the set of chosen items up to a given parameter n, so as to include not only the maximal elements, but the n bigger elements: the relation $more\_urgent\_items(X, Y, now, n)$ holds iff

$$\#Y = n \ \wedge\ \forall\ i[t] \in X.\ i[t] \in Y \ \Leftrightarrow \nexists\ j[t'] \in X \setminus Y.\ j[t'] \rhd i[t],$$

where # is the cardinality of a set. Unfortunately, this does not guarantee anymore TCS satisfiability, for instance in presence of strict temporal constraints, i.e. $<$, between two actions that could be selected together and executed at the same instant.

Example 4 *Let us consider the set of actions* $X = \{a[t_a], b[t_b], c[t_c]\}$ *with the constraints* $\{t_a < t_b, t_b <= 100, t_c <= 100\}$. *It holds*

$$a[t_a] \rhd b[t_b] \qquad a[t_a] \rhd c[t_c]$$

while $b[t_b]$ *and* $c[t_c]$ *are not comparable. We have that* $Y = \{a[t_a]\}$, *for* $more\_urgent\_items(X, Y, now)$, *while* $Y' = \{a[t_a], b[t_b], c[t_c]\}$, *for* $more\_urgent\_items(X, Y', now, n)$ *with* $n \geq 3$. *Note that executing the* Y' *set of action at now does not satisfies the constraint* $t_a < t_b$. *On the other hand, executing* Y *may render* $b[t_b]$ *and* $c[t_c]$ *timed out when now is too close to* 100.

It is worth pointing out that the choice of not imposing any (correct) ordering in the core selection functions has been done so as to guarantee the generality of the model, in accordance with the idea that a computee operating in a *global computing* environment can not be guaranteed to always behave correctly, but rather it is expected to be able to react to its faults or the problems it may encounter in the environment, for instance by revising its failed plans or updating its goals. The choice between the two options and a suitable tuning of n are heuristic design choices, depending on the specific domain the computee is designed for, and the features, e.g. punctuality vs. correctness, the designer is more interested in.

The computational counterpart of the definition for $\rhd$, can be addressed by reasoning over the domains of the time variables. These can be determined by means of decidable procedures, based on constraint (arc-consistency) propagation over finite domains, which are polynomial in time (if inequalities are not allowed). Deadlines are trivially determined from variable domains, where ∞ means an unconstrained domain. Constraint theory is however out of scope here. However, it is possible to adopt a simpler brute force approach, consisting in comparing all the possible pairs of items and hence ordering them.

Definition of the notion of *very urgent* items. In order to define a punctual computee, the notion of *very urgent items* needs to be introduced. It will be used by the punctual profile to prefer an AE or PI transition, whenever there are actions or goals that are very urgent since close to their deadline. Given a temporal constraint store TCS, the constant now indicating the present time, and a time parameter u, an item $i[t]$ is *very urgent* with respect to u iff

$$\models_{\mathcal{R}} TCS \cup \{t = now\} \wedge \not\models_{\mathcal{R}} TCS \cup \{t >= now + u\}$$

i.e. the deadline of the item is at most u time instants close, although the item is not timed out yet. The parameter u is critical for the effectiveness of the profile, since a too loose (big) u makes too many items very urgent, while a too tight one makes items urgent possibly when they are too close to their deadline to be dealt with. The relation $very\_urgent\_items(X, Y, now, u)$ holds iff Y contains all the items of X that are very urgent, i.e.

$$\forall i[t] \in X.\ i[t] \in Y \Leftrightarrow i[t] \text{ is very urgent w.r.t. } u.$$

State characteristics. In the states of an operational trace of a simple punctual computee all the AE or PI transitions are applied to input sets containing the more urgent items amongst those selected by the core selection functions. On the other hand, in the states of a puncutal computee, whenever there are urgent items either an AE or a PI transition has to be applied.

1. Given an operational trace of a simple punctual computee

$$T_0(S_0, Y_0, S_1, \tau_0), \ldots T_i(S_i, Y_i, S_{i+1}, \tau_i), \ldots$$

for all i such that $T_i = AE$ or $T_i = PI$, the set Y_i satisfies the relation $more\_urgent\_items(X_i, Y_i, \tau_i, n)$, (or $more\_urgent\_items(X_i, Y_i, \tau_i)$ according to the design choices of the computee), where X_i is the input set returned by the (core) selection function in the transition T_i.

2. Given an operational trace of a punctual computee

$$T_0(S_0, Y_0, S_1, \tau_0), \ldots T_i(S_i, Y_i, S_{i+1}, \tau_i), \ldots,$$

let VUA_i be the set of very urgent actions in S_i ($very\_urgent\_actions(X_i, VUA_i, now, u)$, with X_i returned by the core action selection function), and VUG_i the set of very urgent goals ($very\_urgent\_goals(X_i, VUG_i, now, u)$, with X_i returned by the core goal selection function). Then for all i

   (a) $VUA_i \neq \emptyset \Rightarrow T_i = AE \vee (VUG_i \neq \emptyset \wedge T_i = PI)$, and
   (b) $VUG_i \neq \emptyset \Rightarrow T_i = PI \vee (VUA_i \neq \emptyset \wedge T_i = AE)$.

   This condition says that whenever there are very urgent actions the transition must be AE, unless there are very urgent goals and the transition is PI, and analogously for the presence of very urgent goals. No priority is fixed amongst AE and PI when both very urgent actions and goals are present.

Cycle theory characterisation. A simple punctual computee can be programmed by means of a basic cycle theory: when planning or executing actions give precedence to to the items that are closer to their deadlines amongst those returned by the core selection functions.

The cycle theory $\mathcal{T}$ of a simple punctual computee contains the following *basic* rules (abstracting away from n instantiation):

$$\begin{array}{rl} Punctual_{T|AE}(S', Y): & \\ *AE(S', Y) \leftarrow & T(S, X, S', \tau), selected\_actions(S', X'), now(\tau'), \\ & more\_urgent\_actions(X', Y, \tau'), Y \neq \emptyset. \end{array}$$

$$\begin{array}{rl} Punctual_{T|PI}(S', Y): & \\ *PI(S', Y) \leftarrow & T(S, X, S', \tau), selected\_goals(S', X'), now(\tau'), \\ & more\_urgent\_goals(X', Y, \tau'), Y \neq \emptyset. \end{array}$$

This schema of rules are the only ones enabling AE or PI transitions after a generic T transition. As for the case of the cautious computee, this behaviour can be further restricted by limiting the transitions that can be followed by an AE or PI transition. This can be obtained by replacing the above schema, with rules where T is instantiated for each rule with the name of one of the transitions that can be followed by AE or PI, and the body of the rule is unchanged.

The cycle theory of a punctual computee consists of the above basic rules and a behavioural theory that imposes the selection of an AE or PI transition whenever there are very urgent items. The *behaviour* rules are as follows.

$$
\begin{array}{ll}
Punctual^{T'}_{AE \succ T}(S', Y): & \\
Punctual_{T'|AE}(S, Y) \succ \mathcal{R}_{T'|T}(S', X) \leftarrow & selected\_actions(S', X'), now(\tau'), \\
& very\_urgent\_actions(X', Y, \tau', u), \\
& Y \neq \emptyset.
\end{array}
$$

$$
\begin{array}{ll}
Punctual^{T'}_{PI \succ T}(S', Y): & \\
Punctual_{T'|PI}(S, Y) \succ \mathcal{R}_{T'|T}(S', X) \leftarrow & selected\_goals(S', X'), now(\tau'), \\
& very\_urgent\_goals(X', Y, \tau', u), \\
& Y \neq \emptyset.
\end{array}
$$

The two rules above state that AE (PI) is preferred to all the other transitions when there are very urgent actions (goals), according to the idea that if there are actions and goals close to their deadlines to require an AE or PI transition is recommendable. It is required that the ones above are the only rules in the punctual cycle theory that define preferences over AE and PI transitions, i.e. another rule $\mathcal{R}^{T'}_{T \succ W}$, where W is (or can be instantiated to) either AE or PI, does not exist in the theory.

Since the cycle theory adopts a *credulous* approach, the lack of a priority amongst AE and PI in presence of both very urgent actions and very urgent goals results in a non-deterministic choice. If one wants to control also this case, it is enough to make the conditions of the above rules disjoint. For instance, in order to have a fixed priority of AE over PI in presence of both very urgent actions and very urgent goals, it is enough to modify the second rule in

$$
\begin{array}{ll}
Punctual^{T'}_{PI \succ T}(S', Y): & \\
Punctual_{T'|PI}(S, Y) \succ \mathcal{R}_{T'|T}(S', X') \leftarrow & selected\_goals(S', X), \\
& selected\_actions(S', W), now(\tau'), \\
& very\_urgent\_goals(X, Y, \tau', u), \\
& very\_urgent\_actions(W, Z, \tau', u), \\
& Y \neq \emptyset, Z = \emptyset.
\end{array}
$$

which imposes that PI can be preferred only when there are not very urgent actions.

Compliance of the cycle theory with the state characteristic for a puncutal computee. Referring to a simple punctual computee, whose only basic rules enabling AE and PI transitions are the ones labeled as *Punctual*, the correspondence between state characteristic and cycle theory is trivial. Similarly to the case of a cautious computee, let us consider an operational trace of a simple punctual computee

$$T_0(S_0, X_0, S_1, \tau_0), \ldots T_i(S_i, X_i, S_{i+1}, \tau_i), \ldots,$$

where for some i it holds $T_i = AE$. Then, the only basic rule enabling the transition is (an instance of) $Puncutal_{T|AE}$ It follows that X_i must satisfy the relation $more\_urgent\_actions(X'_i, X_i, \tau', n)$ and $X_i \neq \emptyset$, (X'_i is the set of actions returned by the core action selection function), as specified by the state characteristic for this profile. The case for $T_i = PI$ is analogous.

Let us now consider the case of a punctual computee, with preference of AE over PI when both very urgent actions and goals are present, i.e. the cycle theory $\mathcal{T}$ contains the *Puncutal* as the only basic rules enabling AE and PI, the $Punctual^{\mathcal{T}'}_{AE \succ T}$ and the emended $Punctual^{\mathcal{T}'}_{PI \succ T}$ behaviour rules (which are required to be the only ones determining a preference over AE and PI).

Under the condition of preference of AE over PI, the state characteristic can be rewritten as

$$(VUA_i \neq \emptyset \Rightarrow T_i = AE) \ \wedge \ (VUA_i = \emptyset \wedge VUG_i \neq \emptyset \Rightarrow T_i = PI)$$

Let us assume that $VUA_i \neq \emptyset$ for a given i, and prove that necessarily $T_i = AE$. We have to prove that exists $\Delta \subseteq \mathcal{T}$ admissible argument which concludes AE. The set Δ_P, consisting of the *Puncutal* basic rules and the $Punctual^{\mathcal{T}'}_{AE \succ T}$ and $Punctual^{\mathcal{T}'}_{PI \succ T}$ behaviour rules, is such an admissible argument that concludes AE. Indeed, AE can be derived by the $Puncutal_{T|AE}$ rule, whose conditions in the body are satisfied and no other argument that attacks Δ_P can be found in $\mathcal{T}$. Indeed, let us assume that a counter argument Δ_A that attacks Δ_P exists. Then Δ_A should conclude an incompatible transition different from AE by means of a different basic rule $\mathcal{R}$. But then, such a rule should be preferred to $Puncutal_{T|AE}$ by a behaviour rule in $\Delta_A \subseteq \mathcal{T}$, but this would violate the condition *"a rule $\mathcal{R}^{\mathcal{T}'}_{T \succ W}$, where W is (or can be instantiated to) either AE or PI, does not exist in $\mathcal{T}$"*.

The case for $VUA_i = \emptyset \ \wedge \ VUG_i \neq \emptyset$ is analogous.

5.3 Punctual profile features

The punctual profile behaviour presents two distinguishing features:

1. it sequences the items in its plan so that those with closer deadlines are dealt with first (simple punctual). This can be formalised as follows:

   Proposition 5.1 *Let PP be a computee with a simple punctual profile, and*

   $$T_0(S_0, Y_0, S_1, \tau_0), \ldots T_i(S_i, Y_i, S_{i+1}, \tau_i), \ldots,$$

   be one of its traces. Then for each i such that $T_i = AE$, if $a[t_a] \in Y_i$ then there is no action $b[t_b]$ in the plans of PP in S_i such that it is nor executed or timed-out and it holds $b[t_b] \rhd a[t_a]$.

   Proof. Trivial, by the definition of $more\_urgent\_actions(X_i, Y_i, \tau_i)$, with X_i returned by the core action selection function in the transition i.

2. according to its state characteristic, as proved above, it "forces" AE or PI whenever there are items arbitrarily close to their deadlines (punctual). This feature may be used to guarantee that actions or goals in the plan do not become timed-out before they have been dealt with. A critical issue is the parameter u, determining the time window in which items are considered critically urgent before their deadlines. Clearly this can not leave out of consideration the costs for planning or executing actions. For instance, assuming that the computee performs an action at the time, u could be dynamically defined in function of the number of actions still to be executed in the plan: if there are n such actions, starting to execute them at least n instants before the closest deadline, guarantees that

all will be executed on time. On the other hand, setting a too big u may render too many actions urgent, possibly unnecessarily forcing the computee to repeating AE, while other transitions could be useful, or needed, to improve its individual welfare.

Both the above behaviours have been observed by the experimentation reported in the following example.

Example 5 *Let us consider a computee that has the goal of going to work before time* 20*. Its plan is to get the bus, but this requires to have a ticket as precondition. The plan for having a ticket is simply to buy one. Its* KB_{plan} *is (analogously its* KB_{TR}*):*

```
initiates(get_bus, _, go_to_work). precondition(get_bus,
have_ticket). initiates(buy_ticket,_,have_ticket).
```

and the representation of its initial state is

```
< [],                                    %% KB_0
  [goal((go_to_work,T), root_1, [])],    %% goals
  [],                                    %% empty plan
  [T#<20] >                              %% temporal constraints
```

In this oversimplified representation, the trace of the normal profile selects actions and goals independently of their temporal constraints. It first (attempts to) gets the bus, and then to buy the ticket, hence failing to achieve its goal that can not be proved to hold according to the final KB_0 *(indeed, the action* `get_bus` *has been executed at* 2*, when the precondition* `have_ticket` *was not enforced by the action* `buy_ticket`*, which has been executed at* 6*):*

```
HISTORY

  --> 0: step(INIT,[])
  --> 1: step(PI,((go_to_wok,eqv(t1)),root_1,[]))
  --> 2: step(AE,((get_bus,eqv(t4)),(go_to_wok,eqv(t1)),[have_ticket],[]))
  --> 3: step(GI,[])
  --> 4: step(RE,[])
  --> 5: step(PI,((have_ticket,eqv(t4)),(go_to_work,eqv(t1)),[]))
  --> 6: step(AE,((buy_ticket,eqv(t8)),(have_ticket,eqv(t4)),[],[]))
  --> 7: step(GI,[])
  --> 8: step(RE,[])
  --> 9: step(GR,[])
  --> 10: step(PR,[])
  --> 11: ...

KB 0
   -> executed(((get_bus,2),(go_to_work,eqv(t1)),[have_ticket],[]))
   -> executed(((buy_ticket,6),(have_ticket,eqv(t4)),[],[]))
```

Instead, the punctual computee, in this implemented version that executes actions when they become very urgent (with a notion of urgency bounded to be 5 *instant close to deadlines), succeeds in achieving its goal.* `go_to_work` *is now entailed by the final* KB_0*, where* `buy_ticket` *has been executed at* 14 *and* `get_bus` *at* 15*, in the correct order and just* 5 *instants before their respective deadlines:*

```
HISTORY

  --> 0: step(INIT,[])
  --> 1: step(PI,((go_to_work,eqv(t1)),root_1,[]))
  --> 2: step(GR,[])
  --> 3: step(PR,[])
  --> 4: step(PI,((have_ticket,eqv(t4)),(go_to_work,eqv(t1)),[]))
  --> 5: step(GR,[])
  --> 6: step(PR,[])
  --> 7: step(RE,[])
  --> 8: step(GR,[])
  --> 9: step(PR,[])
  --> 10: step(RE,[])
  --> 11: step(GR,[])
  --> 12: step(PR,[])
  --> 13: step(RE,[])
  --> 14: step(AE,((buy_ticket,eqv(t8)),(have_ticket,eqv(t4)),[],[]))
  --> 15: step(AE,((get_bus,eqv(t4)),(go_to_work,eqv(t1)),[have_ticket],[]))
  --> 16: step(GI,[])
  --> 17: step(RE,[])
  --> 18: step(GR,[])
  --> 19: step(PR,[])
  --> 20: step(GI,[])
  --> 21: ---

KB 0
   -> executed(((buy_ticket,14),(have_ticket,eqv(t4)),[],[]))
   -> executed(((get_bus,15),(go_to_work,eqv(t1)),[have_ticket],[]))
```

It is also interesting to note how in steps 14 *and* 15 *two consecutive AE, which are not usually admitted in a normal profile, have been forced due to the presence of very urgent actions.*

5.4 Discussion

Intuitive advantages. The main advantage of a punctual computee is not to let goals and actions become not achievable because of their deadlines. Moreover, under the cited assumptions, it induces a correct ordering between goals and actions that are present in its plans. This is intended to facilitate (top) goal satisfaction, to minimise failures due to a not careful management of time or a wrong sequencing of action execution. Not surprisingly, the effectiveness of the profile may highly depend on the specific features of the application domain, on which an optimal tuning of the parameter u may depend. Problems due to a too big u have been already mentioned above, while a too small u, say 0, could probably not allow the computee to manage actions and goals before their deadlines. In this sense, not only the pace of environment changes is relevant but also the computational/execution costs of the computee. Indeed, in the present formulation the costs for action execution or planning have not been taken into account, since in principle a computee could try to execute (plan for) all the urgent actions (goals) in one transition. In this respect a suitable tuning of the other parameter n, i.e. the number of

items dealt with in a transition, could be valuable for the computee effectiveness, intended as a balanced distribution of action execution and planning over time.

Trivially, in the general case, it is possible to have examples where the punctual computee increases its welfare function more than a computee with a different behaviour profile, but also vice-versa. It is expected that in a time-critical or fast evolving environment, the punctual computee performs better than other computees.

Alternative definitions.

- *Impatiently punctual* computee. Given its notion of urgency, the computee gives up all the plans and goals that appear "too urgent" (possibly also considering their complexity). This computee abandons a goal/plan as soon as it realises it can not manage to achieve/execute it on time. This has a relation with the idea of considering costs of action execution and planning above mentioned. The enhanced effectiveness of this profile should be due to not wasting time in pursuing goals that will not be achieved.

6 The Impatient Profile

In this section we examine the impatient profile of behaviour. In section 6.2 we identify the characteristic feature of the impatient profile. In section 6.3 we define what an *impatient cycle theory* is and prove that any such cycle theory induces an operational trace which has the required feature. Finally in section 6.4 we give a new definition of what an *impatient cycle theory* is and we define and prove comparison properties for the impatient profile.

6.1 Informal Definition.

In order to give the informal definition of the impatient profile of behaviour, we first need to give the following definition.

Definition 6.1 *We say that two actions* $A_1 = \langle a_1[t_1], \_, \_\rangle$ *and* $A_2 = \langle a_2[t_2], \_, \_\rangle$ *are of the same type iff* $a_1 = a_2$.

An *impatient* profile of behaviour prevents a computee from executing an action, if an action of the *same type* was executed in the past *unsuccessfully*, that is, without producing the desired effect. A more moderate version of this profile can be obtained by defining a time window after which the computee is allowed to execute the action. In our study of the profile, the computee never executes an action if an action of the same type was executed unsuccessfully in the past, unless no other transition is enabled.

6.2 Formal Definition

Characteristic Feature of the Impatient Profile. Given any transition $T_i(S_{i-1}, X_i, S_i, \tau_i)$ in the operational trace of the computee such that $executed(a[t], \tau_i) \in KB_0^i$ for some action $A = \langle a[t], \_, \_\rangle$, where $S_i = \langle KB^i, Goals^i, Plan^i, TCS^i\rangle$, then

- either $KB^{i-1} \nvDash unsuccessful(a[t'])$
- or the only transitions possible from state S_{i-1} are of the form $AE(S_{i-1}, As)$, where for every action $A = \langle a[t], \_, \_\rangle \in As$, it holds that $KB^{i-1} \vDash unsuccessful(a[t'])$.

Where the predicate $unsuccessful(a[t'])$ is defined appropriately using the Temporal Reasoning capability of the computee and it means that an action $\langle a[t'], \_, \_\rangle$ was executed unsuccessfully in the past. This means that $executed(a[t'], \_) \in KB_0$, but the desired effects of the action do not hold at the current time and nothing happened between the time it was executed and the current time that could make the effects not hold.
In other words the computee does not execute an action if an action of the same type was executed unsuccessfully in the past, unless it has no other choice of transition.

6.3 Impatient Cycle Theories

An impatient cycle theory is *any* cycle theory where

- The following two rules are part of the $\mathcal{T}_{behaviour}$ part of the theory:
  - $Impatient\_\mathcal{P}^T_{T1 \succ AE} : \mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, As)) \leftarrow \exists A \in As, A = \langle a[t], \_, \_\rangle, unsuccessful(a[t'])$
    for every T and $T1, X$ such that either $T1 \neq AE$ or $T1 = AE$ and $X \neq As$.
  - $Impatient\_\mathcal{MP}^T_{T1 \succ AE} : Impatient\_\mathcal{P}^T_{T1 \succ AE} \succ \mathcal{P}^T_{AE \succ T1}$,
    for every T and $T1$, such that $T1 \neq AE$.
- There is no rule which could enable $\mathcal{P}^T_{AE \succ T1} \succ Impatient\_\mathcal{P}^T_{T1 \succ AE}$ in the $\mathcal{T}_{behaviour}$ part of the cycle theory.

The purpose of the rule $Impatient\_\mathcal{P}^T_{T1 \succ AE}$ is to give lower priority over any other transition to transitions of Action Execution with input parameter a set of actions As when an action of the same type as an action in As was executed unsuccessfully in the past. The purpose of the rule $Impatient\_\mathcal{MP}^T_{T1 \succ AE}$ is to give the rule $Impatient\_\mathcal{P}^T_{T1 \succ AE}$ higher priority than any other priority rule which gives higher priority to an AE transition.

Proposition 6.1 *An impatient cycle theory induces an operational trace which has the characteristic feature of the impatient profile.*

Proof In order to prove the proposition we will try to construct an admissible set of rules which concludes an action execution transition for a set of actions As when an action of the same type as an action in As was executed unsuccessfully in the past, assuming that there is at least one other transition enabled. When we fail to do so, we will have proven that it is not an admissible conclusion and therefore the cycle theory satisfies the required property.

Formally we will try to construct an admissible set of rules which concludes the transition $AE(S_i, As)$, when there exists an action $A \in As$, where $A = \langle a[t], \_, \_\rangle$ such that $KB^i \models unsuccessful(a[t'])$, assuming that there exists a possible transition $T'(S_i, X)$, where either $T' \neq AE$ or $T' = AE$, and for every action $A' = \langle a'[t], \_, \_\rangle \in X$, $KB^i \not\models unsuccessful(a'[t'])$.

The set must contain a base rule of the form $\mathcal{R}_{T|AE}(S, As)$ that enables an action execution as the next transition. Let's assume that such a rule exists and is enabled at the current state. Let's first show that a set Δ consisting of only this rule is not admissible. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks.

Let's now try and construct a set Δ' that attacks Δ. Based on our assumption, there exists a possible transition $T'(S_i, X)$, which means that there exists a rule $\mathcal{R}_{T|T'}(S, X)$ which is currently enabled. Let's add this rule to the set Δ' along with the two rules

that were added to the cycle theory in order to achieve the impatient profile: $\Delta' = \{\mathcal{R}_{T|T'}(S,X), Impatient\_\mathcal{P}^T_{T1\succ AE}, Impatient\_\mathcal{MP}^T_{T1\succ AE}\}$.

We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash AE(S,As)$, $\Delta' \vdash T'(S,X)$, where $AE \neq T'$ or $As \neq X$, and $incompatible(AE(S,As), T'(S,X))$ holds. Since there is no rule in Δ that has higher priority than any rule in Δ', Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $\mathcal{R}_{T|T'}(S,X)$ in Δ' has higher priority than the rule $\mathcal{R}_{T|AE}(S,As)$ in Δ according to the rule $Impatient\_\mathcal{P}^T_{T1\succ AE}$ in Δ'. But Δ does not have a rule with higher priority than any rule in Δ'. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

In order to extend the set Δ let's assume that there exists a priority rule in the $\mathcal{T}_{behaviour}$ part of the cycle theory that gives higher priority to the rule $\mathcal{R}_{T|AE}(S,As)$ over any other rule that is currently enabled. The constructed set is $\Delta = \{\mathcal{R}_{T|AE}(S,As), \mathcal{P}^T_{AE\succ T1}\}$. The set Δ is obviously consistent. In order for it to be admissible it must also attack all its attacks. Consider the set $\Delta_2 \subseteq \Delta'$ where $\Delta_2 = \{Impatient\_\mathcal{P}^T_{T1\succ AE}, Impatient\_\mathcal{MP}^T_{T1\succ AE}\}$.

We first need to show that Δ' attacks Δ. The two sets draw opposite conclusions since $\Delta \vdash \mathcal{R}_{T|AE}(S,As) \succ \mathcal{R}_{T|T1}(S,X)$, $\Delta_2 \vdash \mathcal{R}_{T|T1}(S,X) \succ \mathcal{R}_{T|AE}(S,As)$. Since there is no rule in Δ that has higher priority than any rule in Δ_2, Δ' attacks Δ.

In order for Δ to be admissible, it has to attack Δ'. The rule $Impatient\_\mathcal{P}^T_{T1\succ AE}$ in Δ_2 has higher priority than the rule $\mathcal{P}^T_{AE\succ T1}$ in Δ, according to the rule $Impatient\_\mathcal{MP}^T_{T1\succ AE}$ in Δ_2. But Δ does not have a rule with higher priority than any rule in Δ_2. Therefore Δ does not attack Δ', so the set of rules Δ is not admissible.

Note that there is no rule that we could add in the set Δ that would make it admissible, since according to the rule $Impatient\_\mathcal{MP}^T_{T1\succ AE}$ the rule $Impatient\_\mathcal{P}^T_{T1\succ AE}$ has higher priority over any other rule that gives priority to any transition over an action execution transition. Also, according to the cycle theory definition there is no rule in $\mathcal{T}_{behaviour}$ that gives higher priority to any priority rule $\mathcal{P}^T_{AE\succ T1}$ over the rule $Impatient\_\mathcal{P}^T_{T1\succ AE}$.

We demonstrated that there does not exist an admissible set of rules which concludes an action execution transition for a set of actions As when an action of the same type as an action in As was executed unsuccessfully in the past and when there are other transitions enabled. Hence, we can conclude that this transition is not an admissible conclusion of the impatient cycle theory an therefore we have proven that a computee with an impatient cycle theory satisfies the required property.

6.4 Properties of the Impatient profile

We defined the *impatient cycle theory* as any cycle theory with certain rules added and where certain conditions hold. In order to find the properties of the impatient profile we will compare a computee with an impatient cycle theory with a computee with a modified version of the same cycle theory where the extra rules which were added to achieve the impatient behaviour are removed. We will call such a theory an *underlying cycle theory* of the computee.

Proposition 6.2 *A computee with an impatient cycle theory is as good as a computee with an underlying cycle theory with the same knowledge base KB, in the sense that whenever the latter succeeds in achieving a goal, so does the former one, in environments where the value of the preconditions of an action do not change by via some unknown factor in the environment.*

Note that the above proposition holds with the following exception. If the preconditions of an action do not hold when the action is first executed, causing it to fail, but their value is then

set to true as an effect of another action the computee executes, the impatient computee will not try executing an action of the same type as the one that failed, hence the computee with the underlying cycle theory might succeed in achieving a goal which the impatient computee fails to achieve.

This motivates a new definition for the impatient cycle theory, where if the preconditions of an action that was executed unsuccessfully in the past are made to hold by some other action the computee executes, then it should be allowed to execute an action of the same type as the action that failed:

An impatient cycle theory is *any* cycle theory where

- The following rules are part of the $\mathcal{T}_{behaviour}$ part of the theory:
  - The rule $Impatient\_\mathcal{P}^T_{T1 \succ AE}$ which gives lower priority over any other transition to transitions of Action Execution with input parameter a set of actions As if an action of the same type as an action in As was executed unsuccessfully in the past:
    $Impatient\_\mathcal{P}^T_{T1 \succ AE} : \mathcal{R}_{T|T1}(S, X) \succ \mathcal{R}_{T|AE}(S, As) \leftarrow \exists A \in As, A = \langle a[t], \_, \_\rangle, unsuccessful(a[t'])$
    for every T and $T1, X$ such that either $T1 \neq AE$ or $T1 = AE$ and $X \neq As$.
  - The rule $Impatient'\_\mathcal{P}^T_{AE \succ T1}$ which gives higher priority to an action execution transition with input a set of actions As, over some other transition, if an action A' of the same type as an action in the set As was executed unsuccessfully in the past but the values of its preconditions have possibly changed as an effect of some other action that the computee has executed between the time when A' was executed and the current time:
    $Impatient'\_\mathcal{P}^T_{AE \succ T1} : \mathcal{R}_{T|AE}(S, As) \succ \mathcal{R}_{T|T1}(S, X) \leftarrow \exists A \in As, A = \langle a[t], \_, \_\rangle, unsuccessful(a[t']), possible\_prec\_change(S, a)$
    for every T, T_1, X, such that either $T1 \neq AE$ or $T1 = AE$ and $X \neq As$, where the predicate $possible\_prec\_change$ is defined as follows:
    $possible\_prec\_change(S, a) \leftarrow last\_unsuccessful(a[t], \tau_i), \exists B, B = \langle b[t], \_, \_\rangle, executed(b[t], \tau_j), affects(b[t], prec(a)), \tau_j > \tau_i$
    where the predicate $last\_unsuccessful(a[t], \tau)$ is defined appropriately using the Temporal Reasoning capability of the computee and it means that action of type a was executed unsuccessfully at time τ, and the predicate $affects(b[t], prec(a))$ is defined appropriately using the Temporal Reasoning capability of the computee and it means that the execution of the action $b[t]$ can have as an effect the change of the value of at least one of the preconditions of an action of type a.
  - The rule $Impatient'\_\mathcal{MP}^T_{AE \succ T1}$ which gives higher priority to the rule $Impatient'\_\mathcal{P}^T_{AE \succ T1}$ over the rule $Impatient\_\mathcal{P}^T_{T1 \succ AE}$:
    $Impatient'\_\mathcal{MP}^T_{AE \succ T1} : Impatient'\_\mathcal{P}^T_{AE \succ T1} \succ Impatient\_\mathcal{P}^T_{T1 \succ AE}$
    for every T and $T1$, such that $T1 \neq AE$.
  - The rule $Impatient\_\mathcal{MP}^T_{T1 \succ AE}$ which gives higher priority to the rule $Impatient\_\mathcal{P}^T_{T1 \succ AE}$ over any priority rule $\mathcal{P}^T_{AE \succ T1}$ which gives higher priority to an AE transition, when $\mathcal{P}^T_{AE \succ T1} \neq Impatient'\_\mathcal{P}$:
    $Impatient\_\mathcal{MP}^T_{T1 \succ AE} : Impatient\_\mathcal{P}^T_{T1 \succ AE} \succ \mathcal{P}^T_{AE \succ T1}$,
    for every T and $T1$, such that $T1 \neq AE$ and $\mathcal{P} \neq Inpatient'\_\mathcal{P}$.

- There is no rule which could enable $\mathcal{P}^T_{AE \succ T1} \succ Impatient\_\mathcal{P}^T_{T1 \succ AE}$) in the $\mathcal{T}_{behaviour}$ part of the cycle theory, besides the rule $Impatient'\_\mathcal{MP}^T_{AE \succ T1}$.

We will now prove proposition 6.2 based on the above cycle theory.

Proof We want to prove that if there exists an operational trace $T_0, ..., T_\kappa$ induced by the underlying cycle theory such that $S_\kappa \vDash G$ for some goal G, then there exists an operational trace $T'_0, ..., T'_n$ induced by the impatient cycle theory such that $S'_n \vDash G$.

By comparing the underlying cycle theory with the impatient cycle theory we can conclude that the only difference between the operational traces induced by the two cycle theories, is that the impatient computee will not execute an action if another action of the same type was executed unsuccessfully in the past as part of another plan, unless some other action which could affect the preconditions of the action that failed, is executed. On the other hand, an action of the same type as the one that failed might be executed in the operational trace induced by the underlying cycle theory, as part of another plan, even if no action which could affect its preconditions is executed. Hence we conclude that differences in the behaviour of the two computees occur only when actions are executed unsuccessfully.

We have the following cases regarding the operational trace $T_0, ..., T_\kappa$ induced by the underlying cycle theory:

- No action execution in the operational trace fails. In this case, as stated earlier, the operational trace induced by the impatient cycle theory will be identical to the one induced by the underlying cycle theory. This means that $S_\kappa = S'_n$, hence $S'_n \vDash G$ and the proposition holds.

- One action in the operational trace induced by the underlying cycle theory fails. Let $A = \langle a[t], \_, C \rangle$ be that action. Since A is the only action that fails in the operational trace induced by the underlying cycle theory, we can conclude, based on the differences between the two computees, that the operational trace of the computee with the underlying cycle theory is the same as the operational trace of the impatient computee until the time that A is executed. Therefore action A also fails in the operational trace induced by the impatient cycle theory. In this case there exist two subcases. Either some other action is executed after A fails in the operational trace of the computee with the underlying cycle theory which could possibly affect the value of the preconditions of action A, or no such action is executed. Let's study both of these cases:

  - No action that could possibly affect the value of the preconditions of A is executed in the operational trace of the computee with the underlying cycle theory after A fails. In this case if one or more actions of the same type as action A, are executed by the computee with the underlying cycle theory, then they will fail since no action that could make their preconditions hold was executed and since the value of the preconditions could not be changed by the environment. The impatient computee on the other hand, will not try to execute such actions. Obviously the only steps in the operational trace induced by the underlying cycle theory that are not in the operational trace induced by the impatient cycle theory, if any, are some unsuccessful action execution transitions. By ignoring these extra steps we can see that every action in the operational trace of the computee with the underlying cycle theory exists in the operational trace of the impatient computee, and since goal G succeeds

in the operational trace induced by the computee with the underlying cycle theory it will also succeed in the operational trace induced by the impatient computee, therefore the proposition holds.

– An action B, which could affect the value of the preconditions of A is executed some time after action A is executed. In this case, if an action A' of the same type as action A is executed some time after action B is executed, in the operational trace induced by the underlying cycle theory, then action A' will also be executed by the impatient computee. This is true because the impatient cycle theory does not prevent it from executing an action of the same type as an action that was executed unsuccessfully in the past, if some other action which could affect its preconditions is executed in the meantime.

This concludes that every action that succeeds in the operational trace of the computee with the underlying cycle theory also succeeds in the operational trace of the impatient computee, in the case where only one action fails in the operational trace induced by the computee with the underlying cycle theory. Hence in this case, the proposition holds.

- Two or more actions fail in the operational trace induced by the underlying cycle theory. We showed above that when exactly one action fails in the operational trace of the computee with the underlying cycle theory then the only steps in that operational trace of the computee with the underlying cycle theory that are not in the operational trace of the impatient computee are unsuccessful action execution transitions. We can generalize the above and say that the same behaviour occurs when more than one action fails in the operational trace of the computee.

We have proven that every action which is executed successfully in the operational trace induced by the underlying cycle theory is also executed successfully in the operational trace induced by the impatient cycle theory. Hence when a goal is achieved in the operational trace induced by the underlying cycle theory it is also achieved in the operational trace induced by the impatient cycle theory, therefore the proposition holds.

It follows from the above proof that the impatient computee will achieve the same goal as the computee with the underlying cycle theory in the same or less number of steps. This is the case because, as shown above, the impatient computee will not try executing an action of the same type as an action which failed in the past, if no action which could affect its preconditions was executed, whereas the computee with the underlying cycle theory may include such transitions in its operational trace.

Corollary 6.1 *A computee with an impatient cycle theory will achieve the same goal as a computee with the underlying cycle theory in equal or less number of transition steps.*

We can also show that in certain cases a computee with an impatient cycle theory is better than a computee with the underlying cycle theory in that the first succeeds in achieving a goal that the second fails to achieve.

Example 6 *We can easily create a scenario where the impatient computee succeeds but the computee with the underlying cycle theory fails. Suppose that the two computees have exactly the same knowledge base and that the impatient computee executes an action A which belongs*

to a plan for some goal G. Let's say that the execution of A fails. According to the definition of the impatient cycle theory, the impatient computee will not try executing any action of the same type as A unless it has no other choice of transition or some action that could affect its preconditions is executed. Suppose that no such action is executed. Let's say that the computee executes next an action B which belongs to a plan for a goal G' where $G' \neq G$. The execution succeeds and goal G' is achieved.

A possible operational trace for the computee with the underlying cycle theory in the above scenario is to execute action A unsuccessfully and then try executing some other action A' of the same type as A. While the computee with the underlying cycle theory is executing action A', action B times out, hence the computee with the underlying cycle theory fails to achieve goal G'.

Therefore we conclude that in certain scenarios a computee with an impatient cycle theory succeeds where a computee with an underlying cycle theory fails.

7 Careful Profile

7.1 Informal Definition

A computee endowed with a *careful* profile of behaviour examines and revises its current commitments frequently so as to recognise infeasible goals and actions (as well as goals that have been achieved already) as soon as possible. The intuitive advantage of such a behaviour profile would be that the computee's operations are not hindered by superfluous items in the state and that reactive rules will not be triggered unnecessarily by goals/actions that are timed-out and not achieved/executed.

7.2 Formal Definition

This behaviour can be achieved by an operational trace where (at least) every other transition is a State Revision (SR).

Definition 7.1 (Careful profile: transition-based characterisation) *A careful computee is a computee that will never generate an operational trace with two consecutive transitions that are different from SR.*

In fact, this condition is stronger than strictly necessary: As long as there are no redundant or infeasible goals or actions no revision would be required. However, from a pragmatic point of view, Definition 7.1 nevertheless provides us with an appropriate characterisation of careful computees. This is so, because *checking* whether or not a state includes redundant or infeasible goals or actions to be revised is just as costly as performing a state revision in the first place.

7.3 Careful Cycle Theories

Our next goal is to define a cycle theory (or a class of cycle theories) that is guaranteed to induce an operational trace where every other transition is a State Revision. As we shall see this is not as straightforward a goal as it may seem. To illustrate the difficulties and to motivate our choices (which are eventually going to overcome these difficulties), we start by attempting to define a careful cycle theory as an extension of the normal cycle theory.

7.3.1 The normal-careful cycle theory

There are several ways of combining cycle theories (in this case the normal cycle theory with the core rules characterising the careful profile). One option would be to take the union of the two cycle theories (which are sets of basic and behaviour rules) and then, where necessary, to introduce additional behaviour rules that determine the computee's behaviour in case of conflict between the rules stemming from the different parts. Another way, which gives the profile designer less freedom but which results in much simpler cycle theories that can be analysed more easily, would be to work at the level of basic rules as far as possible and to use suitable enabling conditions to control the computee's behaviour. This is the approach we are going to follow here.

To design a careful computee, we need to ensure that basic rules expressing that *SR* should follow any other transition T to get priority over any conflicting rules. Instead of using behaviour rules to this effect, we are simply going to delete such conflicting rules in the first place. Hence, we end up with the following approach:

- **Step 1:** Take the normal cycle theory as a starting point.
- **Step 2:** Remove any basic rules that speak about two consecutive transitions both of which are different from *SR*.
- **Step 3:** Add the following basic rule for each transition T different from *SR:*

$$R_{T|SR}(S') : *SR(S') \leftarrow T(S, X, S', \tau)$$

Note that there cannot be any enabling conditions in this kind of rule; *SR* needs to be enabled under *any* circumstances. Whenever there is already a rule $R_{T|SR}$ in the normal cycle theory, that rule should be overwritten by the new rule.

This approach will render some behaviour rules redundant (because in the new cycle theory there is less potential for conflict). We are going to assume here that such redundant rules get deleted, although it makes no difference at the formal level (in practice, of course, it is certainly better to delete them to reduce the complexity of the preferential reasoning processes). We end up with the following *normal-careful cycle theory:*

- $T_{initial}$ is as for the normal cycle theory, i.e. the first transition will be either *GI* or *PI*, depending on whether the set of goals returned by the goal selection function is empty or not.
- Besides the aforementioned rules of the form $R_{T|SR}$, $\mathcal{T}_{basic}$ also contains the following rules:

$$R_{SR|PI}(S') : *PI(S', Gs) \leftarrow SR(S, S'), Gs = c_{GS}(S', \tau), Gs \neq \{\,\}$$
$$R_{SR|GI}(S') : *GI(S') \leftarrow SR(S, S'), Gs = c_{GS}(S', \tau), Gs = \{\,\}$$

  $\mathcal{T}_{basic}$ will not contain any other rules, because all the remaining basic rules in the normal cycle theory speak about transition that should follow transitions other than *SR* and these are fixed for the careful profile.

- It turns out that also all of the rules in $\mathcal{T}_{behaviour}$ are *redundant*, because they also speak about what to do after a transition other than *SR*.

In summary, the normal-careful cycle theory will force a computee to alternate between *SR* and *PI* or *GI* (depending on whether there are currently goals to plan for). Such a computee would be careful, but not functional. We stress again that this is not a consequence of our particular approach which favours deleting redundant rules rather than introducing additional behaviour rules. A normal-careful cycle theory following such an approach would result in exactly the same computee behaviour.

We conclude that it is not possible to extend the normal cycle theory in a meaningful way to obtain a cycle theory conforming to the careful profile of behaviour. Arguably, this is the case because the strong "global" requirement of having a State Revision as every other transition is very different in nature from the other profiles we have considered, which typically are more "local" in their effects and therefore allow for the construction of a cycle theory based on the normal cycle theory by means of local changes to its rules. It may be possible to overcome these difficulties by extending the language for specifying cycle theories to also allow reference to the *penultimate* transition to guide a computee's behaviour. However, in the sequel we are going to explore how far we can get by using the existing system.

7.3.2 The basic careful cycle theory

Our next aim is to put together the "most basic" cycle theory that conforms to the careful profile and that is not non-functional like the normal-careful cycle theory constructed before.

We are certainly going to require the same basic rules as before to specify that any transition different from *SR* must be followed by *SR*. In addition, every transition (possibly excluding *SR* itself) should be enabled after *SR*, subject to the most basic enabling conditions. We end up with the following *basic careful cycle theory:*

- $\mathcal{T}_{initial}$ is as for the normal cycle theory.
- $\mathcal{T}_{basic}$ contains the following rule for every transition T other than *SR*:

$$R_{T|SR}(S') : *SR(S') \leftarrow T(S, X, S', \tau)$$

  Furthermore, $\mathcal{T}_{basic}$ also contains the following rules:[6]

$$\begin{array}{l} R_{SR|RE}(S') : *RE(S') \leftarrow SR(S, S') \\ R_{SR|PI}(S') : *PI(S', Gs) \leftarrow SR(S, S'), Gs = c_{GS}(S', \tau), Gs \neq \{\,\} \\ R_{SR|GI}(S') : *GI(S') \leftarrow SR(S, S'), Gs = c_{GS}(S', \tau), Gs = \{\,\} \\ R_{SR|AE}(S') : *AE(S', As) \leftarrow SR(S, S'), As = c_{AS}(S', \tau), As \neq \{\,\} \\ R_{SR|SI}(S') : *SI(S', Ps) \leftarrow SR(S, S'), Ps = c_{PS}(S', \tau), Ps \neq \{\,\} \\ R_{SR|AOI}(S') : *AOI(S', As) \leftarrow SR(S, S'), Fs = c_{FS}(S', \tau), Fs \neq \{\,\} \\ R_{SR|POI}(S') : *POI(S') \leftarrow SR(S, S') \end{array}$$

- $\mathcal{T}_{behaviour}$ is empty.

The following proposition states the *correspondence* between the basic careful cycle theory and the (transition-based characterisation of the) careful profile given in Definition 7.1:

Proposition 7.1 (Careful profile) *The basic careful cycle theory induces the careful profile of behaviour: Any computee using the above cycle theory will never generate an operational trace with two consecutive transitions that are different from SR.*

[6] The last of these rules may alternatively be regarded as providing $\mathcal{T}_{interrupt}$.

Proof. This follows immediately from the fact that the basic part of the cycle theory forces a State Revision after every other type of transition (there is exactly one basic rule to determine the follow-up of any transition different from *SR*). □

In the formal model, whenever there is a conflict between basic rules that does not get resolved by means of behaviour rules (as would be the case for our basic careful cycle theory), the next transition will be chosen *nondeterministically*. An appropriate implementation of this feature is important, in particular when this situation can occur frequently. In practice, an implementation strategy that is *fair* in the sense that any enabled transition will be chosen eventually would be appropriate.

7.3.3 Other careful cycle theories

The two careful cycle theories we have considered so far are just two examples; there is a whole range of cycle theories that conform to the careful behaviour profile. As we have argued, our first example, the normal-careful cycle theory is not a useful cycle theory, while our second example, the basic careful cycle theory is, in some sense, the most general cycle theory conforming to the careful profile.

For concrete applications, we may wish to combine the features of careful behaviour with other more specific features. We can construct a careful cycle theory of our choice by taking the basic careful cycle theory as a starting point and then imposing additional behaviour constraints using the following means:

- strengthening the enabling conditions in basic rules that determine the follow-up transition for a State Revision;
- deleting basic rules that determine the follow-up transition for a State Revision;
- adding any kind of behaviour rules;
- deleting rules that have become redundant due to other changes.

Note, however, that we *cannot* add any enabling conditions to the basic rules that state that *SR* has to follow any other transition. Otherwise, the resulting cycle theory cannot be guaranteed to conform to the careful profile of behaviour anymore. We also cannot delete such a rule, unless it has already become redundant due to other changes in the cycle theory. On the other hand, we do have complete freedom with respect to the behaviour rules we might wish to add, because the basic rules never admit any conflict as to what transition to choose after a transition different from *SR* in the first place.

Clearly, any such careful cycle theory will also induce the careful profile of behaviour in the sense of Proposition 7.1.

7.4 A property of careful computees

Informally (under certain circumstances):

1. Careful computees will never generate a reaction via the reactivity transition to timed-out unachieved goals or timed-out unexecuted actions.
2. Careful computees will never generate a reaction via the reactivity transition to actions that may not be timed out yet but which are unexecuted and are no longer necessary.

More formally:

The following will never contribute to the generation of a reaction (i.e. an action in $Plan$ or goal in $Goals$) via the reactivity transition,

1. A timed-out unexecuted action
2. A timed-out unachieved goal
3. An unexecuted action whose execution is no longer needed, i.e. an unexecuted action
   (a) with an ancestor which has already been achieved
   (b) with a sibling that has been timed-out
   (c) with an ancestor which has been time-out,

provided that no action and no goal is timed out between an SR transition and its immediate successor if that is an RE transition.

Proof. Let the assumption that no action and no goal is timed out between an SR transition and its immediate successor if that is an RE transition holds. Suppose a careful computee applies the reactivity transition in a state $S = \langle KB, Goals, Plan, TCS \rangle$.

Then by definition of the careful profile (Definition 7.1), because SR must have been applied in the state immediately prior to S, no action or goal of the type specified in 1–3, above exists in state S. Therefore no such action or goal could possibly contribute to the evaluation of the conditions in the body of any reactive rule in KB_{react}. □

7.5 An example showing the advantage of the careful profile

Here we describe an example informally without using the formal syntax of KGP computees.

Computee C believes that he has registered for a conference $conf05$ but wants not to be registered at the conference. It plans for its goal of not being registered and consequently generates an action in its $Plan$ to cancel his registration at $conf05$. He has a reactive rule in its KB_{react} that says:

> *If (observe that the deadline for cancellation for Conference has reached) and (an action of cancellation of registration at Conference is expected) then tell the bank to stop credit card payment to Conference.*

Suppose before the action of cancellation is executed the computee receives a message from the conference telling him that there was a problem with his initial attempt at registration and so he is not actually registered. So his goal of not being registered is achieved without the need for the cancellation action.

A careful computee will not tell the bank to stop the credit card payment (which is pointless anyway), but, under the same circumstances, a non-careful one might.

8 Focussed Profile

8.1 Informal Definition

In the *focussed* profile of behaviour a computee does not plan for more than one top-level goal at a time. More specifically, a focussed computee remains committed to a goal amongst its top-level goals until

- that goal has been successfully achieved, or
- that goal has become infeasible, or
- that goal is not preferred by the Goal Decision capability anymore, when invoked by the *GI* transition, or
- that goal has an empty plan in the state.[7]

The advantages of the focussed profile come into effect in highly time-critical domains as well as domains where a computee has several goals with mutually incompatible plans. In such situations, a focussed computee can be expected to achieve, at least, some goals, whereby an unfocussed computee may fail completely. This applies, in particular, to computees that have a preference for full planning, i.e. to computees that will compute all the actions in a plan before starting to execute them. By concentrating planning on a single goal at a time, a focussed computee is likely to be faster and it will also avoid wasting computing resources over incompatible plans for other goals.

8.2 Formal Definition

The focussed profile of behaviour has the following characterising property: A focussed computee is a computee that, under no circumstances, will generate an operational trace that includes a state with two distinct top-level goals with children, neither of which is either achieved or infeasible. Here we call a goal G *achieved* in a state if it holds according to the temporal reasoning capability. Furthermore, a goal G is called *feasible* iff neither itself nor any of its children is timed-out. An *infeasible* goal is a goal that is not feasible.

Note that this notion of infeasibility need not persist. A goal G may, at some point, be infeasible, because an action in its current plan is timed-out, but G may again become feasible later on, after the computee has revised its state and computed a new plan. Therefore, the only way to ensure that switching to a new top-level goal for planning is admissible (under the focussed profile) is to first check that infeasible goals will *stay* infeasible. This requires a State Revision. Hence, we can give the following alternative definition of the focussed profile, which is simpler than our first definition.

Definition 8.1 (Focussed profile: state-based characterisation) *A focussed computee is a computee that, under no circumstances, will generate an operational trace that includes a state with two top-level goals with children.*

This definition is stronger (more restrictive) than our first definition, but as argued earlier, it is operationally equivalent to that definition, because a computee can only be sure that switching goals will not violate the focussed profile after having executed a State Revision (or after having performed an analogous check).

8.3 Possible extensions

Note that, according to our definition, focussed computees do not deal with more than one top-level goal at a time, but may switch between top-level goals in some situations, as exemplified by the following example.

[7] The need for this last item will become clear in Example 7.

Example 7 *Consider the following (portion of a) trace:*

$$\ldots, SR(S, X, S', \tau), PI(S', X', S'', \tau'), \ldots$$

with $Top\_Goals(S) = Top\_Goals(S') = Top\_Goals(S'') = \{G_1, G_2\}$. *Assume that* G_1 *already has got a plan in* S*, i.e. the set of items in* $Goals(S) \cup Plan(S)$ *with ancestor* G_1 *is not empty. Assume also that* G_2 *has no plan in* S*, i.e. the set of items in* $Goals(S) \cup Plan(S)$ *with ancestor* G_2 *is empty.*

Suppose that all items in the plan for G_1 *in* S *are timed-out at* τ*, and thus* S' *is such that* $Goals(S') = Top\_Goals(S')$ *and* $Plan(S') = \{\}$. *Suppose also that neither* G_1 *nor* G_2 *are timed-out or achieved at* τ'*, but PI is actually introducing a plan for* G_2*, so that the set of items in* $Goals(S'') \cup Plan(S'')$ *with ancestor* G_2 *is not empty.*

The computee with this trace is focussed according to the above definition. However, it does switch from dealing with goal G_1 *to dealing with goal* G_2*, despite goal* G_1 *being still unachieved and feasible.*

The earlier definition of focussed computee may be modified to prevent goal switching, by comparing successive computee states in traces and force that once a computee has been planning/executing for one top-most level goal in one state, it must stick to that goal in successive states, until the goal has been achieved or has become unachievable. This would amount to getting rid of the last item in the informal description of focussed computee at the beginning of Section 8 (and adding some other suitable conditions instead). This stronger definition of focussed computee would however force extending the notion of cycle theory and operational trace, either by looking at histories of transitions rather than individual transitions when deciding on the next transition, or by introducing additional information into cycle theories, such as variables holding the current top-level goal being dealt with. We therefore leave the stronger definition to future work.

Note also that our notion of focussed computee only refers to *top-level* goals, and not to subgoals or actions. The notion of focussed computee could be extended so as to define computees that are focussed all the way, from top-level goals down.

Finally, note that our definition of focussed computee does not distinguish top-level goals which are reactive and top-level goals which are not. A focussed computee focuses on one single top-level goal at a time, no matter whether this goal is reactive or not.

8.4 Focussed Cycle Theories

To achieve the abstract specification, we have to have a cycle theory that ensures that before any Plan Introduction a State Revision has been performed. This is to ensure that we can proceed with planning for a top-level goal even if some of its current children have become infeasible. However, rather than implementing this behaviour directly, we are simply going to ensure that Plan Introduction is only enabled with respect to a set of goals that a focussed computee may plan for given its current state according to the simplified Definition 8.1.

Definition 8.2 (Focussed cycle theories) *A cycle theory is called focussed iff the initial rule* $R_{0|PI}$ *and the basic rule* $R_{T|PI}$ *for any transition* T *include the enabling condition* $focussed(Gs', S, Gs)$*, where:*

(*i*) *S stands for the current state;*

(*ii*) *Gs' stands for the set of goals that is returned by the goal selection function and Gs stands for the set of goals to which PI will be applied; and*

(*iii*) *the predicate $focussed(Gs', S, Gs)$ succeeds iff $Gs \subseteq Gs'$ and all the goals in Gs are descendants of the same top-level goal (possibly including that top-level goal itself) and no other top-level goal has got any children.*

In other words, $focussed(Gs', S, Gs)$ holds iff $Gs \subseteq Gs'$ and

$$\exists G^t \in Top\_Goals : [(\forall G \in Gs : G = G^t \vee G \in descendents(G^t, Goals)) \wedge (\forall G \in Top\_Goals : G \in Gs \rightarrow G^t = G)]$$

The focussed variant of the normal cycle theory would have in $\mathcal{T}_{initial}$ the rule

$$R_{0|PI}(S_0) : *PI(S_0, Gs) \leftarrow Gs' = c_{GS}(S_0, \tau), focussed(Gs', S_0, Gs), Gs \neq \{\}$$

instead of the original rule

$$R_{0|PI}(S_0) : *PI(S_0, Gs) \leftarrow Gs = c_{GS}(S_0, \tau), Gs \neq \{\}$$

Similarly, the focussed variant of the normal cycle theory would have in $\mathcal{T}_{basic}$ the rule

$$R_{AE|PI}(S', Gs) : *PI(S', Gs) \leftarrow AE(S, As, S'), Gs' = c_{GS}(S', \tau), focussed(Gs', S, Gs), Gs \neq \{\}$$

instead of the original rule

$$R_{AE|PI}(S', Gs) : *PI(S', Gs) \leftarrow AE(S, As, S'), Gs = c_{GS}(S', \tau), Gs \neq \{\}$$

This transformation is an example of restricting $\mathcal{T}_{basic}$ and $\mathcal{T}_{initial}$ by adding extra conditions in the body of their rules.

The *correspondence* between the state-based characterisation of the focussed profile and the class of focussed cycle theories may be stated as follows:

Proposition 8.1 (Focussed profile) *Any cycle theory that is focussed according to Definition 8.2 induces the focussed profile of behaviour according to Definition 8.1.*

Proof. The enabling condition $focussed(Gs', S, Gs)$ restricts the set of goals for which the computee may plan to precisely the set of goals that are available for planning according to the state-based characterisation of the focussed profile. The claimed correspondence then follows immediately from the fact that Plan Introduction is the only transition that can add non-top-level goals to a state. □

8.5 A property of the focussed profile

We have briefly discussed the intuitive advantages of the focussed profile of behaviour at the beginning of Section 8. The following proposition makes these advantages more precise.

Informal description: Let a focussed computee be one equipped with a focussed cycle theory, and a normal computee be one equipped with the normal cycle theory. Then if the two computees have a set of goals for which they have no compatible plans then the focussed computee may be able to achieve at least some of its goals while the normal computee may not be able to achieve any of the goals. The theorem below shows under what conditions the focussed computee is guaranteed to achieve more of its goals compared to the normal computee.

Theorem 8.1 *Let F be a focussed computee and N be a normal computee. Let F and N be in a state $S = \langle KB, Goals, Plan, TCS \rangle$ at time T such that:*

1. *Plan is empty, and*
2. *Goals consists of top level goals $G_1, \ldots, G_n$, $n > 1$. (Note :This state can arise, for example, if F and N have just executed Goal Introduction (GI) starting from the same initial state.)*
3. *The goal selection function, in state S, at all times T', $T' \geq T$, selects the same set of k goals for some $k > 1$. Assume these are $\{G_1, \ldots, G_k\}$, without loss of generality.*

Then if

4. *At all times T', $T' \geq T$, KB, $Plan$, $Goals$, $\{G_1, \ldots, G_k\} \models^{T'}_{plan} \langle G_1, \bot, \bot \rangle, \ldots, \langle G_k, \bot, \bot \rangle$, and*
5. *At all times T', $T' \geq T$, and for all G_i, $i = 1 \ldots k$, KB, $Plan$, $Goals$, $\{G_i\} \models^{T'}_{plan} \langle G_i, As, \emptyset \rangle$, where As is not $\bot$ (i.e. the planning capability produces a full plan successfully)*

then, F will achieve at least one of the goals amongst $G_1, \ldots, G_n$, while N will achieve none of them, i.e.

$$\exists T'.\exists i.i = 1..n, T' > T, G_i = \langle l[t], \_ \rangle : KB_{TR} \models_{tr} holds(l,t) \wedge t = T',\ \models_{\Re} TCS \wedge t = T'$$

in the case of F, and

$$\neg\exists T'.\exists i.i = 1..n, T' > T, G_i = \langle l[t], \_ \rangle : KB_{TR} \models_{tr} holds(l,t) \wedge t = T',\ \models_{\Re} TCS \wedge t = T'$$

in the case of N, provided that:

6. *KB_{react} is empty, and*
7. *No POI, AOI transitions are performed, and no GI transition is performed after the establishment of top-level goals $G_1, \ldots, G_n$, and*
8. *Actions are executed in the right order, and*
9. *Goals and actions are non-time critical, in the sense that no goal or action is ever timed out.*

Note that this theorem shows that under the stated conditions a focussed profile is better in terms of welfare, measured by the number of goals a computee achieves.

Proof. (Sketch) Consider the case of the normal computee N: Because of conditions 3,4,6,7 the state of the computee remains the same (although time progresses). In this state because of conditions 3 and 4 the computee can never make any progress towards achieving any of its top level goals.

Now consider the case of the focussed computee F: At some time T_1, $T_1 \geq T$, F performs Plan Introduction (PI). By conditions 3 and 5 and the definition of the focussed profile a goal G_i, $i = 1, \ldots k$, is selected and PI succeeds in producing a complete plan, and updates its

state by adding all the produced actions As to its $Plan$ and updating TCS appropriately. These new actions will then all be executed. They will never be timed-out by condition 9. So they may be removed from the state of the computee by Plan Revision only if their associated goal is achieved. By condition 8 the actions will be executed in the right order, thus earlier actions establish the preconditions for the later actions. Any new goals and actions that may be introduced by later applications of PI will not interfere with the execution of the actions in As. Therefore, finally, after all the actions are executed, it will be possible to prove by TR that goal G_i which was selected at time T_1 is achieved. □

Note that conditions 1–9 are sufficient but not necessary conditions. For example condition 7 can be replaced with one that requires only that any observation recorded as a result of a POI is "independent" of the goals $G_1, \ldots, G_n$, and allows GI transitions but imposes restrictions on their frequency. It is possible to construct examples where some, possibly many, of conditions 1-9 do not hold, but still the focussed computee performs better in goal achievement terms.

8.6 Experiment

Aim. The aim of this experiment is to provide empirical evidence for the advantages of the focussed profile of behaviour discussed in D13 [**?**]. Recall that a focussed computee would be expected to be more successful than a computee that is not focussed in situations where both computees have (at least) two top-level goals that are mutually exclusive in the sense that executing a plan for the first goal will make any potential plan for the other goals infeasible.

Parameters considered and varied. We compare the behaviour of two computees: computee 1 is focussed, while computee 2 uses the normal cycle theory. In a nutshell, a computee is focussed iff any basic rule in its cycle theory that enables the PI transition has got an enabling condition that only allows for the selection of (i) a top-level goal if no other top-level goals have children and (ii) a non-top-level goal G if the top-level goal G' that is the ancestor of G is the only top-level goal with children.

Knowledge bases. We consider a scenario where both computees aim at achieving the two goals `g1` and `g2`. However, the plans for these two goals are not compatible. The domain-dependent part of KB_{plan} is as follows:

```
initiates(a1,_,g1).  precondition(a1,p).  initiates(a3,_,p).
initiates(a2,_,g2).  precondition(a2,q).  initiates(a4,_,q).
[happens(a1,T1),happens(a4,T2)] implies [false].
[happens(a2,T1),happens(a3,T2)] implies [false].
executable(_).
```

The domain-dependent part of KB_{TR} is defined accordingly. KB_{react} is empty.

Runs executed. We have run both computees, initialising them with the two goals `g1` and `g2`, to be achieved before time `20`. Both computees start by planning for `g1`, generating the action `a1` and the subgoal `p`. Then computee 2 is free to plan either for `g2` or for `p`, while the focussed computee 1 is forced to plan for `p` during the next PI transition. Only planning for `p` will lead to a feasible plan for `g1`, while planning for `g2` causes the integrity constraints in KB_{plan} to fire during future calls to planning (resulting in a failure).

We should stress that computee 2 *could* also plan for `p` and proceed in the same way as computee 1. However, in the concrete implementation, the top-level goal `g2` will be selected before `p` when the implemented action selection function is used. Using the focussed profile, the right choice of goals to plan for is guaranteed, independently from such low-level features of a concrete implementation.

Evaluation. The experiment is intended to demonstrate a situation in which a focussed computee is more successful than other computees. This would suggest that in certain domains, where computees have to choose amongst a number of mutually incompatible goals, the focussed profile of behaviour can aid computees to achieve at least a minimum number of goals in good time. In our experiments, the two computees do indeed exhibit the expected difference in behaviour. However, these experiments have also revealed a problem (with both the formal model and the implementation), which causes the focussed computee to execute actions `a1` and `a3` in the wrong order. This issue is currently under investigation.

9 Objective Profile

In this section we will present a cycle theory that implements the objective profile, we call it here the objective cycle theory. Similar to other cycle theories, the objective cycle theory is an extension of the normal cycle theory discussed before. In doing this, a characteristic feature of the objective profile is identified as a guide in the extension.

9.1 Objective Profile

Informally, a computee with an objective profile always attempts to check immediately after the execution of an action that the desired effects of that action were indeed established. The computee therefore attempts to obtain from the environment explicit information confirming the expected result of the execution of its actions. This profile is suited to volatile environments where exogenous events can interfere with the execution and the result of an action.

9.2 Characteristic Properties of Computee States

This computee profile of trying to confirm the effect of actions is formalised by the following property: Given any transition $T_i(S_{i-1}, X_i, S_i, \tau_i)$, in the operational trace of the computee and T_i is AE and $expected\_effects(S_i, X_i, \tau_i) \neq \emptyset$, then there exists a transition $T_j(S_{j-1}, X_j, S_j, \tau_j)$ in the operational trace such that:

- $j > i$;
- T_j is an Active Observation Introduction, and

$$X_j = expected\_effects(S_i, X_i, \tau_i) = \bigcup_{A=\langle a'[\_], G', \_, \_\rangle \in X_i} Obj(S_i, a', \tau_i) \cup \{G'\} \backslash \{\bot\}$$

  where

$$Obj(S, a, \tau) = \{f |\ S \models^{\tau}_{TR} initiates(a, \tau, f)\} \cup \{\neg f |\ S \models^{\tau}_{TR} terminates(a, \tau, f)\}$$

- for all transitions T_k, $k = i+1, ..., j-1$, T_k is a POI transition.

9.3 Objective Cycle Theories

For this profile we simply need to ensure that any Action Execution transition is followed by an Active Observation Introduction transition for fluents that include the effects of the actions that are executed. Formally, we have

Definition 9.1 (Objective Cycle Theory) *An Objective Cycle Theory $\mathcal{T}^{objective}$ is the normal theory extended with the following rules:*

$$\begin{array}{l}\mathcal{R}_{AE|AOI}(S', X) : *AOI(S', X) \leftarrow AE(S, As, S', \tau), \\ \qquad\qquad\qquad\qquad\qquad\qquad X = expected\_effects(S', As, \tau), X \neq \emptyset. \\ Objective\_\mathcal{P}^{AE}_{AOI \succ T'} : \mathcal{R}_{AE|AOI}(S, X) \succ \mathcal{R}_{AE|T'}(S, Y). \\ Objective\_\mathcal{MP} : Objective\_\mathcal{P}^{AE}_{AOI \succ T} \succ \mathcal{P}^{AE}_{T \succ T'}\end{array}$$

for every $T \neq AOI, POI$.

Notice that the first rule replaces the existing $\mathcal{R}_{AE|AOI}$ in the normal cycle theory.

To prove the cycle theory has the expected objective behaviour, we need the follow lemmas.

Lemma 1 *Let $H_{EE} = expected\_effects(S', X)$, then*

$$\{\mathcal{R}_{AE|AOI}(S', H_{EE}),\ AE(S, X, S')\} \models AOI(S', H_{EE}).$$

Proof trival.

Theorem 9.1 *Let $\mathcal{T}^{objective}$ be the objective cycle theory. Then a computee having cycle theory $\mathcal{T}^{objective}$ will behave objectively. That is, let $H_{EE} = expected\_effects(S', A)$, then*

$$\mathcal{T}^{objective} \wedge AE(S, X, S') \models_{pr} AOI(S', H_{EE}).$$

Proof Let $\triangle = \{\mathcal{R}_{AE|AOI}(S', H_{EE}), Objective\_\mathcal{P}^{AE}_{AOI \succ T}, Objective\_\mathcal{MP}\}$, we show that $\triangle$ is an admissible argument. Let $\triangle'$ attack to $\triangle$, then by defintion, there must exist $\mathcal{R}' \in \triangle'$ and $\mathcal{R} \in \triangle$ such that

$$\triangle' \models \mathcal{R}' \succ \mathcal{R}.$$

But $\mathcal{R}$ can neither be $Objective\_\mathcal{P}^{AE}_{AOI \succ T}$ nor $Objective\_\mathcal{MP}$ because $\mathcal{T}^{objective}$ does not contain such $\succ$ rules. Therefore it must be that $\mathcal{R} = \mathcal{R}_{AE|AOI}(S', H_{EE})$. As there are only three such $\succ$ rules in $\mathcal{T}^{objective}$, that is,

$$\begin{array}{l}\mathcal{P}^{AE}_{AE \succ T} : \mathcal{R}_{AE|AE}(S, As) \succ \mathcal{R}_{AE|T}(S, X) \\ \mathcal{P}^{AE}_{GR \succ T} : \mathcal{R}_{AE|AOI}(S, As) \succ \mathcal{R}_{AE|T}(S, X) \\ \mathcal{P}^{AE}_{AOI \succ T} : \mathcal{R}_{AE|GR}(S, As) \succ \mathcal{R}_{AE|T}(S, X)\end{array}$$

to attack $\triangle$, $\triangle'$ must contain either $\mathcal{P}^{AE}_{AE \succ T}$ or $\mathcal{P}^{AE}_{GR \succ T}$. However the rule $Objective\_\mathcal{P}^{AE}_{AOI \succ T}$ in $\triangle$ has higher priority than both of these two rules, according to the rule $Objective\_\mathcal{MP}$ in $\triangle$. But $\mathcal{T}^{objective}$ contains no rule with higher priority than $Objective\_\mathcal{MP}$, so does $\triangle'$. Therefore, $\triangle'$ can not attack $\triangle$. This concludes that $\triangle$ is an admissible set.

By lemma and above argument, we have

$$\triangle \models_{pr} AOI(S', H_{EE}).$$

Note that objective profile generally does not improve the welfare of individual computees compared with normal profile and in some case it is even worse. For example, given a computee with an initial state consisting of only

- one top level goal $g(\tau)$ and
- one plan $\{\langle a_1[\tau_1], g, []\rangle, \langle a_2[\tau_2], g, []\rangle\}$ and
- $TCS = \{\tau_2 = \tau_1 + 1, \tau = \tau_2 + 1\}$.

Due to the temporal constraints, the goal can be achieved only in the case that the two actions are executed one by one. This can happen for a computee having normal profile but is impossible for a copmputee with objective profile.

However, in some cases, objective profile does have advantages over the normal profile as demonstrated by the following example.

Example of advantages suppose a computee C has a top level goal $g(\tau)$ and there are two plans P_1 and P_2 for g, with

$$P_1 = \{\langle a_1[\tau_1], g, []\rangle, \langle a_2[\tau_2], g, [q]\rangle\}, \quad P_2 = \{\langle a[\tau_0], g, []\rangle$$

and in P_1, a_1 has an expected effect, q, which, in turn, is a pre-condition of a_2. The formalization is as follows.

- **State**
  - KB_{plan}:
    $initiates(a, T, g)$
    $precondition(a, true)$
    $initiates(a_1, T, q)$
    $initiates(a_2, T, g)$
    $precondition(a_2, q)$
    $executable(a)$
    $executable(a_1)$
    $executable(a_2)$
  - $KB_0 = [\ ]$
  - $Goals = \{G = \langle g[T], \bot\rangle\}$
  - $Plan = \{\langle a_1[T_1], g[T], []\rangle, \langle a_2[T2], g[T], [q]\rangle\}$
  - $TCS = \{T_1 < 6, T_2 = 9, T < 10\}$
- **Behaviour**
  1. If C operates under the normal profile, then C will behave as follows:
     - ...,

- by AE, a_1 is executed, say, at time 5,
- ...,
- by AE, a_2 is executed at time 9,
- via SR, a_1, a_2 are removed, and in revising G , C finds that a_1 has not met the expected effect, and therefore G is not satisfied. However, C now can not introduce P_2 because G has to be removed due to it is now timee-out(at time 10).

2. However, if C operates under the objective profile, then C will behave as follows:
   - by AE, a_1 is executed, say, at time 5,
   - via AOI at time 6, C finds that a_1 fails to meet its expected effect and $observed(\neg q, 6)$ is added to current state. This results in that a_2 can not be selected for execution because its pre-condition is false.
   - C then goes to SR at time 7, a_1, a_2 are removed from state, while G is kept there because it is not satisfied and not timed-out.
   - by PI, plan P_2 is introduced at time 8.
   - by AE, a is executed at time 9.
   - via AOI at time 10, C finds a's expected effect has been reached.
   - C goes to SR and removes a and g with g being satisfied

10 Related work

Profiles of computees are an attempt to address the need that different applications require different deliberation processes which therefore should be controlled by the designer/programmer. In 3APL [2] a meta-language that refers to goals, actions, plans and rules together with constructs from an imperative programming is used in order to implement how to process the object-level entities in the application.

As mentioned in deliverable D8 [3], unlike KGP which is based purely on declarative logic programming, the 3APL language is a combination of imperative and logic programming. From the imperative programming viewpoint, 3APL inherits the full range of regular programming constructs, including recursive procedures and state-based computation. States of agents in 3APL, however, are belief (or knowledge) bases, which are not characterised by the usual variable assignments of imperative programming. From the computational logic perspective, also taken by our KGP model, answers to queries in the beliefs of a 3APL agent are proofs in the logic programming sense.

Regarding attitudes of agents, most of the existing work refers to behaviour at the level of social attitudes and personalities whereas our cycle profiles refer more to the level of problem solving strategies (given an application environment). For example, in [1] five profiles (agreeable, disagreeable, argumentative, open-minded, elephant child) are proposed to discriminate between different attitudes of agents with varying degree of willingness to cooperate.

In a computee this aspect of its personal attitude can be captured as a part of its Goal Decision knowledge base where it can influence its decisions (see e.g. [8, 7]). Nevertheless the two levels are linked as a specific personal attitude can lead or induce a problem solving strategy. In [10] different agent architectures are studied where each one of these is based on a different combination of reactive/deliberative capabilities depending on the personal attitude to its current needs. An different survival behaviour emerges from the different architectures.

In our computees such variety of behaviour can be achieved via a combination of suitable Goal Decision and Cycle theories.

Another stream of research is based on the quantitative evaluations of agent behaviours. One of more measures of the behaviour are given and then agents are either evaluated on practical test-bed experimentations, or studied under a given probabilistic distribution of environment changes and actions performed by agents. These attempts, from an empirical or probabilistic viewpoint, also try to provide guidelines about the adequacy of classes of behaviours with respect to classes of environments.

For instance, in [9], agent behaviour is seen as the capability to transform conditions that hold in the environment, then measures as *power*, *greatest potential*, *usefulness*, *reliability*, etc., are defined as a mean to compare behaviours, and the relationships between this measures are studied, e.g. the greatest potential is bigger for more powerful behaviours. This developed theory is then used to sustain (empirical or probabilistic) guidelines for the development and engineering of agents, like *include behaviours with weak preconditions and strong consequences to improve usefulness.*

These kind of work, although based on different settings, share similarities and aims with our profile-based approach. On the long term, a diffused practice of profiles-based agent engineering will facilitate the identification of those profiles that have been demonstrated more useful, effective, economical, well-fare increasing etc, with respect to the environment of interest.

11 Conclusions

This reports summarises the progress on the specification of properties of computees based upon their operational trace and induced by a cycle theory. We have formally defined various cycle theories corresponding to various computee profiles of behaviour and we have stated and proved certain properties of these.

The study of profiles of behaviour seems promising as we have demonstrated that based on the KGP model we can program in a declarative and modular way the required pattern of behaviour. While studying comparison properties between the different profiles, the need for formally defining suitable notions of the environment became obvious. More work is needed in this area, which will help in defining and proving more profile related properties.

References

[1] L. Amgoud and S. Parsons. Agent dialogue with conflicting preferences. In *Proceedings of ATAL01*, 2001.

[2] M. Dastani, F. S. de Boer, F. Dignum, W. van der Hoek, M. Kroese, and J. C. Meyer. Programming the deliberation cycle of cognitive robots. In *Proc. of 3rd International Cognitive Robotics Workshop (CogRob2002)*, Edmonton, Alberta, Canada, 2002.

[3] M. Gavanelli, E. Lamma, P. Torroni, P. Mello, K. Stathis, P. Moraïtis, A. C. Kakas, N. Demetriou, G. Terreni, P. Mancarella, A. Bracciali, F. Toni, F. Sadri, and U. Endriss. Computational model for computees and societies of computees. Technical report, SOCS Consortium, 2003. Deliverable D8.

[4] A. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. A logic-based approach to model computees. Technical report, SOCS Consortium, 2003. Deliverable D4.

[5] A. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. Declarative agent control. Technical report, SOCS Consortium, 2004.

[6] A. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. The KGP model of agency. Aug. 2004.

[7] A. Kakas, N. Maudet, and P. Moraitis. Layered strategies and protocols for argumentation-based agent interaction. In *Proceedings of ArgMA04*, 2004.

[8] A. Kakas, P. Torroni, and N. Demetriou. Agent planning, negotiation, and control of operation. In *Proceedings of ECAI04, pp. 28-32*, 2004.

[9] A. D. Mali. On the evaluation of agent behaviours. *Artificial Intelligence*, 147:1–17, 2003.

[10] M. Scheutz, A. Sloman, and B. Logan. Emotional states and realistic agent behaviour, 2000.

A Normal cycle theory

This is a modification of the *normal cycle theory* given in D8[3], where the two transitions GR and PR are replaced by a single transition SR (for State Revision) and some additional rules are incorporated in the basic part. Note that the notation given in D8 is used.

The normal cycle theory specifyies a pattern of operation where the computee prefers to follow a sequence of transitions that allows it to achieve its goals in a way that matches an expected "normal" behaviour. Basically, the "normal" computee first introduces goals (if it has none to start with) via GI, then reacts to them, via RE, and then repeats the process of planning for them, via PI, executing (part of) the chosen plans, via AE, revising its state, via SR, until all goals are dealt with (succesfully or revised away). At this point the computeee returns to introducing new goals via GI and repeating the process above. Whenever in this process the computee is interrupted via a passive observation, via POI, it chooses to introduce new goals via GI, to take into account any changes in the world. Whenever it has actions which are "unreliable", in the sense that their preconditions definitely need to be checked, the computee senses them (via SI) before executing the action. Whenever it has actions which are "unreliable", in the sense that their effects definitely need to be checked, the computee actively introduces actions that aim at sensing these effects, via AOI, after having executed the original actions.

- $\mathcal{T}_{initial}$ consists of

  $r_{0|GI}(S_0) : GI(S_0) \leftarrow empty\_goals(S_0)$

  $r_{0|PI}(S_0) : PI(S_0, Gs) \leftarrow Gs = c_{GS}(S_0, \tau), Gs \neq \{\}$

  $r_{0|POI}(S_0) \leftarrow poi\_pending(\tau)$

  This last rule is only needed with the generalisation of operational trace given in D8, if we want to enforce that passive observations are always taken into account when occurring (as in the original notion of operational trace in D4).

- $\mathcal{T}_{basic}$ consists of the following parts.
  - The rules for deciding what might follow an AE transition are as follows:
    $r_{AE|PI}(S', Gs) : PI(S', Gs) \leftarrow AE(S, As, S'), Gs = c_{GS}(S', \tau), Gs \neq \{\}$
    $r_{AE|AE}(S', As') : AE(S', As') \leftarrow AE(S, As, S'), As' = c_{AS}(S', \tau), As' \neq \{\}$
    $r_{AE|AOI}(S', Fs) : AOI(S', Fs) \leftarrow AE(S, As, S'), Fs = c_{FS}(S', \tau), Fs \neq \{\}$
    $r_{AE|SR}(S') : SR(S') \leftarrow AE(S, S')$
    $r_{AE|GI}(S') : GI(S') \leftarrow AE(S, S')$
    Namely, AE could be followed by another AE, or by a PI, or by an AOI, or by an SR, or by a GI, or by a POI. Any other possibility, e.g. for SR to follow AE, is excluded within this particular $\mathcal{T}_{basic}$ theory.
  - The rules for deciding what might follow SR are as follows
    $r_{SR|PI}(S') : PI(S', Gs) \leftarrow SR(S, S'), Gs = c_{GS}(S', \tau), Gs \neq \{\}$
    $r_{SR|GI}(S') : GI(S') \leftarrow SR(S, S'), Gs = c_{GS}(S', \tau), Gs = \{\}$
    Namely, SR can only be followed by PI or GI, depending on whether there are goals to plan for or not in the state. 360,1 29
  - The rules for deciding what might follow PI are as follows
    $r_{PI|AE}(S', As) : AE(S', As) \leftarrow PI(S, Gs, S'), As = c_{AS}(S', \tau), As \neq \{\}$
    $r_{PI|SI}(S', Ps) : SI(S', Ps) \leftarrow PI(S, Gs, S'), Ps = c_{PS}(S', \tau), Ps \neq \{\}$
    The second rule is here to allow the possibility of sensing the preconditions of an action before its execution.
  - The rules for deciding what might follow GI are as follows
    $r_{GI|RE}(S', Gs) : RE(S') \leftarrow GI(S, S')$
    $r_{GI|PI}(S', Gs) : PI(S', Gs) \leftarrow GI(S, S'), Gs = c_{GS}(S', \tau), Gs \neq \{\}$
    Namely, GI can only be followed by RE or PI, if there are goals to plan for.
  - The rules for deciding what might follow RE are as follows
    $r_{RE|PI}(S', Gs) : PI(S', Gs) \leftarrow RE(S, S'), Gs = c_{GS}(S', \tau), Gs \neq \{\}$
    $r_{RE|SI}(S', Ps) : SI(S', Ps) \leftarrow RE(S, S'), Ps = c_{PS}(S', \tau), Ps \neq \{\}$
    ADDED by ALEX/WENJIN:
    $r_{RE|AE}(S', As) : AE(S', As) \leftarrow RE(S, S'), As = c_{AS}(S', \tau), As \neq \{\}$
    $r_{RE|SR}(S') : SR(S') \leftarrow RE(S, S')$
  - The rules for deciding what might follow SI are as follows
    $r_{SI|AE}(S', As) : AE(S', As) \leftarrow SI(S, Ps, S'), As = c_{AS}(S', \tau), As \neq \{\}$
  - The rules for deciding what might follow AOI are as follows
    $r_{AOI|AE}(S', As) : AE(S', As) \leftarrow AOI(S, Fs, S'), As = c_{AS}(S', \tau), As \neq \{\}$
    $r_{AOI|SR}(S') : SR(S') \leftarrow AOI(S, Fs, S')$
    $r_{AOI|SI}(S', Ps) : SI(S', Ps) \leftarrow AOI(S, Fs, S')Ps = c_{PS}(S', \tau), Ps \neq \{\}$
  - The rules for deciding when POI should take place are as follows
    $r_{T|POI}(S') : POI(S', \tau) \leftarrow T(S, X, S', \tau', \tau), poi\_pending(\tau),$
    for all transitions T, namely POI is always an option if there is an input from the environment (observation) pending. These rules are needed only for the generalised

version of operational trace given in D8, if we want to enforce that passive observations are always taken into account when occurring (as in the original notion of operational trace in D4).

- $\mathcal{T}_{interrupt}$ consists of the following rules

  $r_{POI|GI}(S') : GI(S') \leftarrow POI(S, S')$

  $r_{POI|RE}(S') : RE(S') \leftarrow POI(S, S')$

  $r_{POI|SR}(S') : SR(S') \leftarrow POI(S, S')$

  $r_{POI|POI}(S') : POI(S') \leftarrow POI(S, S')$

  This last rule is only needed with the generalisation of operational trace given in D8, if we want to enforce that passive observations are always taken into account when occurring (as in the original notion of operational trace in D4).

- $\mathcal{T}_{behaviour}$ consists of the following rules (besides its auxiliary part, including the definitions for *incompatible* given above as well as any other definitions for predicates used in $\mathcal{T}_{behaviour}$, such as $empty\_goals, unreliable\_pre$ etc):

  - GI should be given higher priority if there are no goals in *Goals* and actions in *Plan* in the state: [8] $R^{T}_{GI|T'} : h\_p(r_{T|GI}(S), r_{T|T'}(S, X)) \leftarrow empty\_goals(S), empty\_plan(S)$

    for all transitions T, T', $T' \neq GI$, and with T possibly 0 (indicating that if there are no goals and plans in the initial state of a computee, then GI should be its first transition).

  - GI is also given higher priority after a POI:

    $R^{POI}_{GI|T} : h\_p(r_{POI|GI}(S'), r_{POI|T}(S, S'))$

    for all transitions $T \neq GI$.

  - After GI, the transition RE should be given higher priority:

    $R^{GI}_{RE|T} : h\_p(r_{GI|RE}(S), r_{GI|T}(S, X))$

    for all transitions $T \neq RE$.

  - After RE, the transition PI should be given higher priority:

    $R^{RE}_{PI|T} : h\_p(r_{RE|PI}(S, Gs), r_{RE|T}(S, X))$

    for all transitions $T \neq PI$;

  - After PI, the transition AE should be given higher priority, unless there are actions in the actions selected for execution whose preconditions are "unreliable" and need checking, in which case SI will be given higher priority:

    $R^{PI}_{AE|T} : h\_p(r_{PI|AE}(S, As), r_{PI|T}(S, X)) \leftarrow not\, unreliable\_pre(As)$

    for all transitions $T \neq AE$;

    $R^{PI}_{SI|T} : h\_p(r_{PI|SI}(S, Ps), r_{PI|T}(S, As)) \leftarrow unreliable\_pre(As)$

    for all transitions $T \neq SI$;

    Here we assume that the auxiliary part of $\mathcal{T}_{cycle}$ specifies whether a given set of actions contains any "unreliable" action, in the sense described above.

[8] Instead of the conditions $empty\_goals(S), empty\_plan(S)$ in this rule, we could have these conditions in each rule in $\mathcal{T}_{basic}$ which indicates as viable any transition that could be a competitor of GI after any given transition.

– After SI, the transition AE should be given higher priority

$$R^{SI}_{AE|T} : h\_p(r_{SI|AE}(S, As), r_{SI|T}(S, X))$$

for all transitions $T \neq AE$.

– After AE, the transition AE should be given higher priority until there are no more actions to execute in *Plan*, in which case either AOI or SR should be given higher priority, depending on whether there are actions which are "unreliable", in the sense that their effects need checking, or not:

$$R^{AE}_{AE|T} : h\_p(r_{AE|AE}(S, As), r_{AE|T}(S, X))$$

for all transitions $T \neq AE$. Note that, by definition of $\mathcal{T}_{basic}$, the transition AE is applicable only if there are still actions to be executed in the state.

$$R^{AE}_{AOI|T} : h\_p(r_{AE|AOI}(S, Fs), r_{AE|T}(S, X)) \leftarrow BC^{AE}_{AOI|T}(S, Fs)$$

for all transitions $T \neq AOI$, where the behaviour condition $BC^{AE}_{AOI|T}(S, Fs)$ is defined (in the auxiliary part) by:

$$BC^{AE}_{AOI|T}(S, FS) \leftarrow empty\_executable\_plan(S), unreliable\_post(S)$$

Similarly, we have:

$$R^{AE}_{SR|T} : h\_p(r_{AE|SR}(S), r_{AE|T}(S, X)) \leftarrow BC^{AE}_{SR|T}(S)$$

for all transitions $T \neq SR$ where:

$$BC^{AE}_{SR|T}(S) \leftarrow empty\_executable\_plan(S), not\, unreliable\_post(S)$$

Here, we assume that the auxiliary part of $\mathcal{T}_{cycle}$ specifies whether a given set of actions contains any "unreliable" action, in the sense expressed by *unreliable_post*, and defines the predicate *empty_executable_plan*. Intuitively, *empty_executable_plan*(S) succeeds if all the actions which can be selected for execution have already been executed.

– After SR, the transition PI should have higher priority:

$$R^{SR}_{PI|T} : h\_p(r_{SR|PI}(S, Gs), r_{SR|T}(S))$$

for all transitions $T \neq PI$. Note that, by definition of $\mathcal{T}_{basic}$, the transition PI is applicable only if there are still goals to plan for in the state. If there are no actions and goals left in the state, then rule $R^{T}_{GI|T'}$ would apply.

– After any transition, POI is preferred over all other transitions:

$$R^{T}_{PI|T'} : h\_p(r_{T|POI}(S), r_{T|T'}(S, X))$$

for all transitions T, T', $T' \neq POI$, with T possibly 0 (indicating that, if applicable by $\mathcal{T}_{initial}$, POI should be the first transition).

This priority rule is only needed with the generalisation of operational trace given in D8, if we want to enforce that passive observations are always taken into account when occurring (as in the original notion of operational trace in D4).

– In the initial state, PI should be given higher priority:

$$R^{0}_{PI|T} : h\_p(r_{0|PI}(S, Gs), r_{0|T}(S, X))$$

for all transitions $T \neq PI$. Note that PI, by definition of $\mathcal{T}_{initial}$, the transition PI is applicable initially only if there are goals to plan for in the initial state.