SOCS

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

IST-2001-32530

Coherence Properties of the KGP Model

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/ICSTM/74/IN/I/b1
File name:	174-b1[coherence].pdf
Editor:	F. Sadri
Contributing partners:	ICSTM
Contributing workpackages:	WP5
Estimated person months:	N/A
Date of completion:	8 March 2005
Date of delivery to the EC:	18 March 2005
Number of pages:	9

ABSTRACT
In this document we introduce, formalise and prove a number of Coherence Properties. These properties are aimed at demonstrating some beneficial consequences of some of the design decisions made in the declarative and computational models of computees.

Coherence Properties of the KGP Model

F. Sadri and F. Toni

Department of Computing, Imperial College London
Email: {fs,ft}@doc.ic.ac.uk

ABSTRACT
In this document we introduce, formalise and prove a number of Coherence Properties. These properties are aimed at demonstrating some beneficial consequences of some of the design decisions made in the declarative and computational models of computees.

Contents

1 Introduction

The computee declarative model consists of several knowledge base modules, used within a number of capabilities, which, in turn, are used to define transitions. The transitions are integrated within cycle theories and enabled by a collection of four core selection functions. These are Action Selection Function, c_{AS}, Goal Selection Function, c_{GS}, Fluent Selection Function, c_{FS}, and Precondition Selection Function, c_{PS}. For example a cycle step that may result in the Action Execution being chosen as the next transition to be performed relies on the Action Selection Function to select a set of actions from the state of the computee to be executed at that given time and in that given state. Similarly a cycle step that may result in the Plan Introduction transition being chosen as the next transition to be performed relies on the Goal Selection Function to select a set of goals from the state of the computee to be planned for at that given time and in that given state.

In this document we explore some of the consequences of the design details of the computee model, including the definitions of its Action and Goal Selection Functions. This study, as well as showing some of the benefits of the design, also highlighted some errors in the original definition of Action Selection as reported in deliverable D8 [3]. The errors have now been addressed and two alternative approaches to correcting them have been explored. These are reported in the body of Deliverable D13.

The rest of this document is structured as follows. In Section 2 we first recall the definitions of (corrected) Action Selection Function and Goal Selection Function and then briefly review the aspects of cycle theories that are needed in this paper. Then in Section 3 we formalise a number of Coherence properties and prove them and in Section 4 we conclude.

2 Background

The propositions in Section 3 refer to the Goal and Action Selection Functions and cycle theories. Here we briefly recall these.

2.1 Action Selection Function

The Action Selection Function selects actions from the computee state for execution.

Informally, the set of conditions for the (core) Action Selection Function, c_{AS}, is as follows. Given a state $S = \langle KB, Goals, Plan, TCS \rangle$ and a time-point τ, the set of all actions selected by c_{AS} is defined as follows. Let $\mathcal{X}(S, \tau)$ be the set of all actions A in $Plan$ such that:

1. A is executable at τ, i.e. it is not timed out,
2. no ancestor or sibling of A in $Goals$ and $Plan$ is timed out at τ,
3. no ancestor of A in $Goals$ is already satisfied at τ, given S,
4. no precondition of A is known to be false at τ, given S,
5. A has not already been executed.

Then $c_{AS}(S,\tau) \subseteq \mathcal{X}(S,\tau)$ such that all actions in $c_{AS}(S,\tau)$ are executable concurrently at τ.

The above is formalised as follows:

Formally, given a state $S = \langle KB, Goals, Plan, TCS\rangle$, and a time-point τ, the set of all actions selected by c_{AS} is defined as follows. Each action A in $Plan$ is represented as $\langle a[t], G, C\rangle$ where a is the action operator, t is the time associated with the action, G is the immediate goal for which the action has been planned (G is the parent of A), and C is the set of all the preconditions of A. Let $\mathcal{X}(S,\tau)$ be the set of all actions

$$A = \langle a[t], G, C\rangle \in Plan$$

such that:

1. there exists a total valuation σ for the variables in TCS such that $\sigma \models_{\Re} t = \tau \wedge TCS \wedge \Sigma(S)$,
2. there exists no action $\langle a'[t'], G^*, C'\rangle \in Plan$ and there exists no goal $\langle l[t'], G^*\rangle \in Goals$ such that
   - $G^* = G$ or $G^* \in ancestors(G, Goals)$, and
   - there exists no total valuation σ for the variables in TCS such that $\sigma \models_{\Re} t' \geq \tau \wedge TCS \wedge \Sigma(S)$,
3. there exists no $\langle l[t'], G^*\rangle \in Goals$ such that
   - $G^* = G$ or $G^* \in ancestors(G, Goals)$, and
   - there exists a total valuation σ for the variables in TCS such that $\sigma \models_{\Re} t' \leq \tau \wedge TCS \wedge \Sigma(S)$ and $KB \models_{TR} l[t']\sigma$,
4. let $C = l_1[t] \wedge \ldots \wedge l_n[t]$; if $n > 0$, then it is not the case that for some $i = 1, \ldots, n$ there exists a total valuation σ for the variables in TCS such that $\sigma \models_{\Re} TCS \wedge t = \tau \wedge \Sigma(S)$ and $KB \models_{TR} \overline{l_i[t]}\sigma$,
5. $executed(a[t], t') \notin KB_0$.

Then, $c_{AS}(S,\tau) \subseteq \mathcal{X}(S,\tau)$, and $c_{AS}(S,\tau) = \{\langle a_1[t_1], \_, \_\rangle, \ldots, \langle a_m[t_m], \_, \_\rangle\}$ (where $m \geq 0$), such that there exists a total valuation σ for the variables in TCS such that

$$\sigma \models_{\Re} TCS \wedge t_1 = \tau \wedge t_m = \tau \wedge \Sigma(S).$$

2.2 Goal Selection Function

The Goal Selection Function selects goals to be planned for from the computee state.

Informally, the set of conditions for the (core) Goal Selection Function, c_{GS}, is as follows. Given a state $S = \langle KB, Goals, Plan, TCS\rangle$ and a time-point τ, the set of all goals selected by c_{GS} is the set of all goals G in $Goals$ such that at time τ:

1. G is not timed out at τ,

2. no ancestor or sibling of G in *Goals* or *Plan* is timed out at τ,

3. no ancestor of G in *Goals* is satisfied in S at τ,

4. G is not satisfied in S at τ.

Formally, given a state $S = \langle KB, Goals, Plan, TCS \rangle$ and a time-point τ, the set of all goals selected by c_{GS} is the set of all goals

$$G = \langle l[t], G' \rangle \in Goals$$

such that:

1. there exists a total valuation σ for the variables in TCS such that $\sigma \models_{\Re} t > \tau \wedge TCS$,

2. there exists no $\langle a[t'], G', C \rangle \in Plan$, and there exists no $\langle l'[t'], G^* \rangle \in Goals$ such that
   - $G^* = G'$ or $G^* \in ancestors(G', Goals)$, and
   - there exists no total valuation σ for the variables in TCS such that $\sigma \models_{\Re} t' \geq \tau \wedge TCS$,

3. there exists no $G^* = \langle l'[t'], \_ \rangle \in Goals$ such that
   - $G^* \in ancestors(G, Goals)$, and
   - there exists a total valuation σ for the variables in TCS such that $\sigma \models_{\Re} t' \leq \tau \wedge TCS$ and $KB \models_{TR} l'[t']\sigma$,

4. there exists no total valuation σ for the variables in TCS such that $\sigma \models_{\Re} t' \leq \tau \wedge TCS$ and $KB \models_{TR} l[t]\sigma$.

2.3 Cycle theories

In the computee model the control component of computees is specified by cycle theories. These have been discussed in deliverable D12 [1] and in another annex to D13 [2]. For the purposes of the results in the next section it is sufficient to recall the cycle step rules in the normal cycle theory that enable the Plan Introduction (PI) and Action Execution (AE) Transitions. These are the rules that make use of the Action Selection and Goal Selection functions.

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

Note that for the AE transition it is always the case that Action Selection is part of one of the enabling conditions. Similarly, for the PI transition Goal Selection is always part of one of the enabling conditions. These are also the case in all the behaviour profiles that we have defined (see [2]), i.e. all the cycle step rules that enable AE have amongst their conditions a condition of the form $As' = c_{AS}(S', \tau)$, and all the cycle step rules that enable PI have amongst their conditions a condition of the form $Gs = c_{GS}(S', \tau)$.

3 Coherence Properties

In the following propositions we explore some of the consequences of the design of computees. The first proposition shows that computees do not attempt to execute actions that they believe are infeasible or unnecessary, and computees do not attempt to plan for goals if a plan is not needed or if it is too late to plan for them. These properties show the suitability of the design of the KGP model in that computees are prevented from spending time on the activities of planning and acting if these activities are not appropriate or useful.

Proposition 3.1 (Coherence of Plan Introduction and Action Execution)

1. *Computees never attempt to execute actions that*
   - *are timed out, or*
   - *belong to a plan for a goal that is timed out or that they believe is already achieved.*
2. *Computees never attempt to plan for a goal that*
   - *is timed out or that they believe is already achieved, or*
   - *belongs to a plan for a goal that is timed out or that they believe is already achieved.*

Here, given a state, by plan for a goal *we mean the set of all actions and goals that are descendants of that goal within the tree in that state.*

Sketch of the Proof. Any cycle step rule showing that a transition may be followed by Action Execution uses the Action Selection Function in an enabling condition, and any cycle step rule showing that a transition may be followed by Plan Introduction uses the Goal Selection Function in an enabling condition. The required properties above follow from the definition of these two selection functions, as follows.

The first part of the proposition follows directly from the first three conditions of the definition of the Action Selection Function (Section 2.1). The second part of proposition follows directly from the four conditions of the Goal Selection Function (Section 2.2). □

The second proposition, below, shows that computees will avoid concurrent execution of actions that they believe cannot be successfully executed together, because of the temporal constraints imposed upon them.

Proposition 3.2 (Coherence of Action Execution with Temporal Incompatibility) *If for some time τ and actions in a state S with operators $a_1[t_1], \ldots, a_n[t_n]$ there exists no total valuation of the variables $t_1, \ldots, t_n$ satisfying the temporal constraints in S, all equalities derived from KB_0 in S ($\Sigma(S)$) and $t_1 = \tau, \ldots, t_n = \tau$, then all of $a_1[t_1], \ldots, a_n[t_n]$ will never be selected together for execution at τ.*

Proof. This property follows directly from the last requirement specified in the definition of the Action Selection Function which, in effect, states that actions that are temporally incompatible, given the constraint on their times, will not be selected for execution at the same time. □

Note that it was while considering this proposition a mistake in the definition of the Action Selection Function was identified and corrected.

The third proposition below show that computees will never try to execute two actions at the same time if the actions have incompatible preconditions. This result uses both the declarative and computational models of computees and shows that their design is such that computees will avoid concurrent execution of actions that they believe cannot be successfully executed together.

Proposition 3.3 (Coherence of Action Execution with Incompatible Preconditions) *Given the declarative and computational models of computees [3] if*

- *the $Plan^{nr}$ part of the computee state contains two actions $A1$ and $A2$ (i.e. $A1$ and $A2$ are non-reactive actions), and*
- *$A1$ has a precondition $p1$ and $A2$ has a precondition $p2$ such that the computee's KB_{plan} includes an integrity constraint*
  IC
  assume_holds_at($p1$,T) and assume_holds_at($p2$,T) implies $false$

then the computee will never attempt to execute actions $A1$ and $A2$ at the same time.

Proof. Actions A1 and A2 are non-reactive actions. So they must have been generated in the computee state by a PI transition. The declarative and computational models of the computee ensure that the same PI transition which generates actions $A1$ and $A2$ will also generate a constraint $T1 \neq T2$ in the TCS part of the computee state where $T1$ and $T2$ are the proposed execution times of actions $A1$ and $A2$, respectively. The temporal constraint $T1 \neq T2$ is generated because the Planning Capability first generates subgoals assume_holds_at($p1$,$T1$) and assume_holds_at($p2$,$T2$) which then together with the IC, above, generate the temporal constraint.

Then the proposition follows from Proposition 2. □

4 Conclusion

In this document we have explored some useful consequences of a number of the detailed design decisions made in the declarative and computational models of computees. The three propositions discussed show that the design ensures that, no matter which profile of behaviour is chosen, computees will not spend time pursuing timed-out goals and actions, or generating unnecessary plans, or attempting to execute actions concurrently when the concurrent execution will not succeed according to the computee's own beliefs.

This study has also helped identify some shortcomings in the design which we have addressed. An example has been the definition of the Action Selection Function which was identified as incorrect. We report a correction in the body of D13.

To the best of our knowledge no similar properties have been proven or identified in the literature.

Our investigation of Coherence Properties can be extended in at least two directions. One is to identify and prove other properties. For example one property that is very likely to hold is that computees will never generate a plan for a goal if they believe that the plan will not succeed, for example because of some foreknowledge about the expected behaviour of other computees or the environment. Another property worth exploring is the extent to which the Reactivity Transition lends itself to plan repair [4]. Reactivity allows computees to adapt to the changes in their environment by setting themselves additional goals to achieve and extra actions to execute, and, as such, part of the functionality of Reactivity can be seen as adapting the computees' plans to the changing environment.

Another very interesting direction is to explore useful properties that do not hold in the current design and to identify how the model can be modified to ensure such properties. One such example concerns the Planning Capability. It would be interesting to see how it can be modified to ensure that the plans it generates are ”minimal”, possibly in the sense of requiring as few actions as possible, or in the sense of minimising any dependence on other computees to execute actions. Another useful topic could be to see how the Goal Introduction Capability can be modified so that it does not completely over-write the state but exploits as much of the state as possible, for example by re-using as much of the partial plans already generated.

References

[1] M. Alberti, A. Bracciali, F. Chesani, U. Endriss, M. Gavanelli, A. Guerri, A. Kakas, E. Lamma, P. Mancarella, P. Mello, M. Milano, F. Riguzzi, F. Sadri, K. Stathis, G. Terreni, F. Toni, and P. Torroni. Update report on WP1–WP6. Technical report, SOCS Consortium, 2004. Deliverable D12.

[2] F. Athienitou, A. Bracciali, U. Endriss, A. Kakas, W. Lu, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. Profile-related properties. Technical report, SOCS Consortium, 2005. Annex to deliverable D13.

[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] B. Nebel and J. Koehler. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artrificial Intelligence*, 76(1–2):427–454, 1995.