Knowledge-Based Asynchronous Programming

Hendrik Wietze de Haan Wim H. Hesselink
Gerard R. Renardel de Lavalette

Department of Mathematics and Computing Science, University of Groningen,
P.O. Box 800, 9700 AV Groningen, The Netherlands
{hwh,wim,grl}@cs.rug.nl

The full version of this paper will appear in *Fundamenta Informaticae*.

A knowledge-based program (KBP) is a description of the behaviour of agents in terms of knowledge that an agent must have before (s)he may perform an action. The definition of the meaning (semantics) of KBPs is problematic, since it involves a vicious circle; on the one hand, the meaning of a KBP depends on the meaning of the knowledge operators for the agents involved, on the other hand, the meaning of the knowledge operators depends on the collection of possible executions of the KBP, i.e. on the meaning of the KBP. In the standard approach, given by Fagin, Halpern, Moses and Vardi in their book *Reasoning about Knowledge* (1995), the semantics of a KBP is a collection of runs (sequences of states), *implicitly* defined by the composition of protocols (not involving knowledge) via synchronous parallellism. The implicit definition avoids the circularity mentioned above, but uniqueness is not guaranteed.

Our approach aims at an explicit and unique definition of the sematics of KBPs. It is based on the observation that the implicit definition of the meaning of a KBP can be read as a fixed point of an automorphism. We work this out as follows.

A KBP is a non-deterministic choice of guarded commands: $P = (\varphi_1 \to f_1) \cup \ldots \cup (\varphi_n \to f_n)$, where the φ_i are epistemic formulae and the $f_i : X \to X$ are actions on the state space X. The intended meaning of guarded command $\varphi_i \to f_i$ is: if φ_i holds, then perform action f_i. More formally: given parameter $B \subseteq X^+$, a collection of traces, we define the interpretation $[\![\varphi]\!]_B \subseteq X^+$ of formulae φ and $[\![P]\!]_B \subseteq (X^+ \times X^+)$ of programs P. Here $B \subseteq X^+$, the collection of possible traces, is a parameter required for the interpretation of knowledge operators:

$$[\![K_i\varphi]\!]_B = \{xs \in X^+ \mid \forall\, ys \in B(xs \approx_i ys \;\Rightarrow\; ys \in [\![\varphi]\!]_B)\}$$

where $xs \approx_i ys$ means: agent i cannot distinguish trace xs from ys. In general $xs \in [\![\varphi]\!]_B$ iff φ holds in xs, and $(xs, ys) \in [\![P]\!]_B$ iff P leads from xs to the extension ys of xs.

We define the automorphism $F : \mathcal{P}(X^+) \to \mathcal{P}(X^+)$ by $F(B) = X_0[\![P]\!]_B$, the collection of traces in $[\![P]\!]_B$ that start in some initial state $x_0 \in X_0$. Now we define the *unique* meaning of P as an iteration of F, in such a way that it allows as much well-justified knowledge as possible without introducing contradictory knowledge. If F is monotonic, this iteration is the greatest fixed point of F.

Moreover, our approach is based on asynchronous parallellism, which is more realistic in the context of multi-agent systems than synchronous parallellism, using an interleaving semantics with possible delay via *stutterings*. A stuttering of a trace is obtained by repeating certain elements: e.g., *abbbcddeee* is a stuttering of *abcde*. Two traces are stuttering-equivalent if they are both stutterings of a third trace. The definition of our indistinguishability relation $\approx_i$ is based on both an indistinguishability relation of states, and the stuttering equivalence.

The Unexpected Hanging Paradox

We justify our definition of the semantics by a number of examples, including the *Unexpected Hanging Paradox*. A convicted prisoner is to be executed at noon within seven days, but the judge tells him that he will not know the day of his execution, even on the beginning of that day itself. The prisoner might then reason that he cannot be executed on day 7, because when he would still be alive at the beginning of day 7, he would know that he would be executed that day. By backward induction he might reason that he cannot be executed without knowing that he will be executed. Yet, on e.g. day 3, the prisoner is surprised to meet the executioner.

Let us formulate this situation as a KBP. The state space is

$$X = \{(t, n, d) \mid t \in N, n \in \{1, \ldots, 7\}, d \in \{\mathit{true}, \mathit{false}\}\}$$

with initially $t = 1$, n arbitrary and $d = \mathit{false}$. The prisoner p can observe only t and d. The program is

$$(t \neq n \rightarrow t := t + 1) \cup ((t = n \wedge \neg K_p(t = n) \wedge \neg d) \rightarrow (d := \mathit{true}; t := t + 1))$$

If $n < 7$, the program has an execution with d being set to *true*. If $n = 7$ however, the prisoner will know this at $t = 7$, and the program deadlocks. In our approach, deadlock is modeled by forced stutterings. Since no fairness assumptions are given (so there is no guarantee that time progresses), the prisoner cannot use backward induction to exclude execution sequences that lead to deadlock, and this resolves the paradox. This is in agreement with the analysis, based on dynamic epistemic logic, by Gerbrandy in his Ph.D. thesis *Bisimulations on Planet Kripke* (1999).

Directions for future research

We have not found a suitable subclass of KBPs for which the automorphism F is monotonic. It would be useful to study what conditions, if any, would guarantee monotonicity of F.

In our current framework we do not consider temporal formulas. No fairness assumptions or progress properties can be expressed. It would be interesting to extend the framework to infinite traces (runs) and progress properties.

Another direction of research would be to study refinement relations between KBPs in our framework. A KBP can then gradually be refined to a knowledge-free program. This would eliminate the need to explicitly calculate an iteration sequence.