International Joint Conference on Qualitative and Quantitative Practical Reasoning, ECSQUARU-FAPR'97, June 9–12, 1997, Bad Honnef, Germany
Springer LNCS 1244, pp. 436–450

Process Modeling with Different Qualities of Knowledge

Wolfgang May*

Institut für Informatik, Universität Freiburg
Am Flughafen 17, 79110 Freiburg, Germany
may@informatik.uni-freiburg.de

Abstract. For modeling structured processes with different levels of atomic actions, classical linear or branching time Kripke structures with first-order states are insufficient: They do not provide any means for modeling independent parallel threads of activity which have to be joined at some point, action refinement, or procedure concepts. In all three cases, it is important to distinguish facts resp. knowledge derived in the current thread of activity from facts which are not concerned in this activity. This problem is also closely related to the frame problem.
In this paper, hierarchical Kripke structures are introduced for modeling hierarchically structured processes, also coping with the different qualities of knowledge arising in this context: every state consists of a total first-order interpretation, which gives the state as-is, and a partial first-order interpretation containing all procedure knowledge which has been derived in the current thread of activity.
A correct and complete set of axioms is presented for reasoning about hierarchical Kripke structures.

1 Introduction

Variants of Kripke structures are widely used as a model-theoretic base for reasoning about processes, for instance with state-oriented logics as CTL or Dynamic Logic [Har79]. Furthermore, the more process-oriented Hennessy-Milner Logic [HM85] is interpreted over Kripke structures. In some contexts, for example Transaction Logic [BK94], logics refer to path-structures which also can be regarded as variants of Kripke structures.

In any case, modeling independent parallel threads of activity which are rejoined at some synchronization point causes problems. Additionally, action abstraction or refinement is not supported in those models. The problems arising by those features are closely related to the frame problem [MH69]: At the beginning of a (sub)process, there is no local knowledge, the state as-is is the same as the state where the process has been started from. Every thread of activity reads and derives its own facts, making up its local knowledge about the global process. By carrying out actions, a process gains local knowledge from examining the state as-is and modifying it. When joining different threads of activity, the local knowledge after the join consists of the local knowledge of the predecessor state modified by the accumulated knowledge of all incoming threads. The state

* Supported by grant no. GRK 184/1-97 of the Deutsche Forschungsgemeinschaft.

as-is is obtained from the predecessor state, overwritten by the sum of the local knowledge of all participating threads.

From the model-theoretic point of view, the problem is solved by modeling different qualities of knowledge in the states: every state consists of a total first-order interpretation, giving the state as-is, and a partial first-order interpretation containing all procedure knowledge which has been derived in the current thread of activity.

The paper is structured as follows: This section closes with a short glance on related work on structured process modeling. Section 2 briefly reviews of first-order Kripke structures. In Section 3, the algebra of actions used in this paper is introduced. In a first step, in Section 4, the modeling of parallel actions is introduced which is extended in Section 5 with procedures to the complete scope of hierarchical processes. Section 6 contains an illustrating example, Section 7 closes with some concluding remarks.

Related Work. In [BK94], a concept for defining transactions as sequences of elementary actions in a logic-programming style is defined, based on the model-theoretic notion of *path structures*. States and transitions are represented by an abstract concept of a theory and a transition oracle which can be instantiated to any specific formal framework. Parallelism is modeled via interleaving.

From the process-algebraic point of view, action refinement has been dealt with in [DG91, DG95, AH93]. In those works, only the dynamic aspects are considered, thus there is no notion of *states*.

In [AH96], an abstract framework for reactive modules is presented which permits parallel composition and abstraction from the internal behaviour of modules. There, the focus is on the observable behaviour of communication variables, the transitions performed by composed modules are given in a declarative style, similar to the transition oracle of [BK94].

From the software engineering point of view, the concept can further be extended with the usual modularization concepts of import, export, visible signature, renaming, and hiding (cf. [BHK90]).

Specialized instances of the concept have been used for modeling nested transactions in the state-oriented deductive database language Statelog [LML96], and for extending the Evolving Algebra specification method [May97]. Here, the concept is presented from the model-theoretic point of view in its full generality.

2 First-Order Kripke Structures

Every language of first-order logic includes the symbols) and (, the boolean connectives $\neg$ and $\vee$, the quantifier $\exists$, and an infinite set of variables $\mathsf{Var} := \{x_1, x_2, \ldots\}$. A particular language is given by its *signature* Σ consisting of function symbols and predicate symbols with fixed arities $\mathsf{ord}(f)$ resp. $\mathsf{ord}(p)$. Terms, first-order formulas, and the notions of bound and free variables are defined as usual.

A *first-order interpretation* $\mathfrak{I} = (I, \mathfrak{U})$ over a signature Σ consists of a non-empty set $\mathfrak{U}$ (*universe*) and a mapping I which maps every function symbol $f \in \Sigma$ to a function $I(f) : \mathfrak{U}^{\mathsf{ord}(f)} \to \mathfrak{U}$ and every predicate symbol $p \in \Sigma$ to a function $I(p) : \mathfrak{U}^{\mathsf{ord}(f)} \to \{\mathrm{T}, \mathrm{F}\}$. For *partial* interpretations, $I(f)$ and $I(p)$ are partial functions/mappings. The set of interpretations (partial interpretations) over a fixed signature Σ is denoted by $\mathbb{I}_\Sigma$ ($\mathbb{I}_\Sigma^p$).

Definition 1 A first-order *Kripke structure* over a signature Σ is a triple $\mathfrak{K} = (\mathfrak{G}, \mathfrak{R}, \mathfrak{M})$ where $\mathfrak{G}$ is a set of states, $\mathfrak{R} \subseteq \mathfrak{G} \times \mathfrak{G}$ an *accessibility relation*, and for every $g \in \mathfrak{G}$, $\mathfrak{M}(g) = (M(g), \mathfrak{U}(g))$ is a first-order interpretation of Σ with universe $\mathfrak{U}(g)$. $\mathfrak{G}$ and $\mathfrak{R}$ are called the *frame* of $\mathfrak{K}$. In some cases, for a set $\mathfrak{X}$, the accessibility relation $\mathfrak{R} \subseteq \mathfrak{G} \times \mathfrak{X} \times \mathfrak{G}$ is labeled[2].

A *path* p in a Kripke structure $\mathfrak{K} = (\mathfrak{G}, \mathfrak{R}, \mathfrak{M})$ is a sequence $p = (g_0, g_1, g_2, \ldots)$, $g_i \in \mathfrak{G}$ with $\mathfrak{R}(g_i, g_{i+1})$ holding for all i. □

3 The Action Algebra

Allowing composite actions, the set $\mathbb{A}$ of actions is given as an algebra, based on a set $\Sigma_\mathbb{A}$ of action symbols (analogous to function and predicate symbols), the underlying data universe $\mathfrak{U}$ and a set $\mathsf{op}_\mathbb{A}$ of operators for composing actions.

Since for modeling distributed processes, parallelism is a crucial point and modeling of parallelism by interleaving is not always satisfactory, a truly parallel composition is needed. *Parallel composition* of actions is denoted by $\|$. $\in_\|$ denotes membership in parallel composition: For an elementary action a, $\mathsf{a} \in_\| \mathsf{b} \;:\Leftrightarrow\; \mathsf{a} = \mathsf{b} \vee (\exists \mathsf{c}, \mathsf{d} : \mathsf{b} = (\mathsf{c} \| \mathsf{d}) \wedge (\mathsf{a} \in_\| \mathsf{c} \vee \mathsf{a} \in_\| \mathsf{d}))$, i.e. if b is already elementary, or a is a parallel component of some parallel component of b.

Actions consisting of different alternative subactions which can be executed in a single step are constructed by the *alternative composition* $\vee$.

Although sequential composition is modeled naturally by successive steps, it should be possible to combine actions by sequential composition which should be executed atomically in a single step. *Sequential composition* to an *atomically executing* action is denoted by $\otimes$.

For iterative computations, *finite iteration* a^ω and *iterative closure* $\mathsf{a}^\star$ of an action a are defined. On the respective level of abstraction, a^ω and $\mathsf{a}^\star$ are regarded as atomic actions.

Also, for implementing abstraction and action refinement, procedure definitions are provided, altogether summarized in the following definition:

Definition 2

- For an n-ary action symbol a which does not occur on the left side of a procedure definition (see third item of this definition) and $u_1, \ldots, u_n \in \mathfrak{U}$, $a(u_1, \ldots, u_n)$ is an *elementary action*.
- For actions $\mathsf{a}, \mathsf{a}_1, \mathsf{a}_2$, $\mathsf{a}_1 \| \mathsf{a}_2$, $\mathsf{a}_1 \otimes \mathsf{a}_2$, $\mathsf{a} \vee \mathsf{b}$, a^ω and $\mathsf{a}^\star$ are composite actions.
- For an action a with arguments $(u_1, \ldots, u_n)$ and an action symbol p, $\mathsf{p} = p(u_1, \ldots, u_n) := \mathsf{a}$, is a *procedure definition*. p is called a *procedure*.

Since procedure symbols are also action symbols, procedures are also actions, thus, $\mathbb{A}$ is a non-free algebra. With this, action refinement via procedure definitions fits easily into the concept: from an abstract point of view, an action

[2] cf. Dynamic Logic [Har79], Hennessy-Milner Logic [HM85], or Transaction Logic [BK94]; in these cases, $\mathfrak{X}$ corresponds to a set of actions.

symbol p is used for an elementary action, whereas from a more detailed level, it is used as a procedure symbol.

The sets of elementary resp. composite actions are denoted by $\mathbb{A}_e$ resp. $\mathbb{A}_c$, the set of procedures is denoted by $\mathbb{A}_p$. □

Definition 3 For an interpretation $\mathfrak{I}$ and an action a, the state reached by executing a in $\mathfrak{I}$ is denoted by $\mathsf{a}(\mathfrak{I})$. For nondeterministic a, $\mathsf{a}(\mathfrak{I})$ is set-valued. □

3.1 Specification of Actions

The modeling of actions directly leads to the frame problem [MH69]: it has to be specified, which facts – dependent on the current state – *must* hold in the state reached by execution of a particular action, and which facts are simply not effected. The former are reflected in the specification of the action, whereas the latter are assumed to remain unchanged which is normally incorporated into frame axioms. Thus, the *specification* of an action *alone* should not give a *total* interpretation of the successor state – since then there would be no room for parallel composition. Instead, the specification formalism should talk about *partial* interpretations, which can be combined easily.

For composing (partial) interpretations, semantic counterparts to terms and atomic formulas are needed:

Definition 4 Given an interpretation $\mathfrak{I}$ with universe $\mathfrak{U}$ over a signature Σ, a symbol $v \in \Sigma$ and $u_1, \ldots, u_{\mathsf{ord}(v)} \in \mathfrak{U}$, the tuple $(v; u_1, \ldots, u_{\mathsf{ord}(v)})$ is called a *function application* over Σ and $\mathfrak{U}$ if v is a function symbol, and it is called a *predicate application* if v is a predicate symbol. □

Definition 5 The (semantical) *domain* of a (partial) interpretation $\mathfrak{I} = (I, \mathfrak{U})$ over a signature Σ is the (mathematical) domain of the mapping I:

$$\begin{aligned}\mathsf{dom}(\mathfrak{I}) := \{P \mid\ & P = (p; u_1, \ldots, u_m) \text{ a predicate application over } \Sigma \text{ and } \mathfrak{U} \text{ and} \\ & (I(p))(u_1, \ldots, u_m) \text{ is defined}\} \ \cup \\ \{F \mid\ & F = (f; u_1, \ldots, u_n) \text{ a function application over } \Sigma \text{ and } \mathfrak{U} \text{ and} \\ & (I(f))(u_1, \ldots, u_n) \text{ is defined}\} \quad .\end{aligned}$$
□

Note, that these sets are subsets of the Herbrand base and the Herbrand universe.

Definition 6 In the following, let all interpretations be over a fixed signature Σ and universe $\mathfrak{U}$. Let $\sqcup$ denote the least upper bound of two elements in the flat lattices given by ($\bot < u < \top$ for all $u \in \mathfrak{U}$) resp. ($\bot < t < \top$ for $t \in \{\mathrm{T}, \mathrm{F}\}$). For two (partial) interpretations $\mathfrak{I}$ and $\mathfrak{I}'$, the *union* $\mathfrak{J} := \mathfrak{I} \cup \mathfrak{I}'$ is defined as

$$\begin{aligned}(J(p))(u_1, \ldots, u_{\mathsf{ord}(p)}) &:= \sqcup((I(p))(u_1, \ldots, u_{\mathsf{ord}(p)}), (I'(p))(u_1, \ldots, u_{\mathsf{ord}(p)})) \\ (J(f))(u_1, \ldots, u_{\mathsf{ord}(f)}) &:= \sqcup((I(f))(u_1, \ldots, u_{\mathsf{ord}(f)}), (I'(f))(u_1, \ldots, u_{\mathsf{ord}(f)})) \quad .\end{aligned}$$

$\mathfrak{J} = \mathfrak{I} \cup \mathfrak{I}'$ is *consistent*, if $J(p)(\bar{u}) \neq \top$ and $J(f)(\bar{u}) \neq \top$ for all $\bar{u} \in \mathfrak{U}^\omega$. In this case, $\mathfrak{J}$ is a partial interpretation over Σ with universe $\mathfrak{U}$.

For an interpretation $\mathfrak{I}$ over Σ and sets F of function applications and P of predicate applications over Σ and $\mathfrak{U}$, the restriction $\mathfrak{I}|_R = (I|_R, \mathfrak{U})$ of $\mathfrak{I}$ to $R := F \cup P$ is defined as

$$(I|_R\ (p))(u_1, \ldots, u_n) := \begin{cases} (I(p))(u_1, \ldots, u_n) & \text{if } (p; u_1, \ldots, u_n) \in P \\ \bot & \text{otherwise, and} \end{cases} ,$$

$$(I|_R\ (f))(u_1, \ldots, u_n) := \begin{cases} (I(f))(u_1, \ldots, u_n) & \text{if } (f; u_1, \ldots, u_n) \in F \\ \bot & \text{otherwise} \end{cases} ,\quad .$$