Variations of Model Checking

David N. Jansen

FMOODS 2002 Poster

1 Combining Actions And Real-Time

This work is finished and published [7]. A reviewed version of this article will appear shortly.

The logic **ATCTL** is a convenient logic to specify properties with actions and real-time. It is intended as a property language for Lightweight UML models [12], which consist mainly of simplified class diagrams and statecharts.

ATCTL combines two known extensions of CTL, namely ACTL and TCTL. The reason to extend CTL with both actions and real time is that in LUML state–transition diagrams, we specify states, actions and real time, and our properties refer to all of these elements. The analyst therefore needs a property language that contains constructs for all these elements.

ATCTL can be reduced to ACTL as well as to TCTL, and therefore also to CTL. This gives us a choice of tools for model checking; we have used is Kronos [13], a TCTL model checker.

$$\begin{tikzcd}
& \text{ATCTL [8]} & \\
\text{ACTL [4]} \arrow[ur] & & \text{TCTL [1]} \arrow[ul] \\
& \text{CTL [5]} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

1.1 Example: Internal Travel Office

Imagine an *internal travel office* in a large firm, e. g. a university. Employees of the firm book business trips via the office. To do this, the office needs a travel permit and a payment allowance.

As an extra service, also private trips can be booked. The cost of a private trip is deducted from the employee's salary, and no payment allowance is needed.

The office proposes a trip schedule to the employee; as soon as he accepts it, the office tries to book a trip and a hotel for the appropriate period. If this succeeds, the employee is invited to come along and to pick up the necessary documents. If the booking fails, the employee is informed and the trip is cancelled; however, the employee is allowed to restart the procedure. The financial department handles the payments and, if applicable, the deduction from the salary.

We plan to introduce a workflow system that controls this process. We would like to ensure that no payment is made without allowance, and model checking is the method we use.

The example is adapted from Verbeek et al. [11].

The model consists of the lightweight class diagram in figure 1. The two classes' behaviour is shown in the statecharts of figures 2 and 3. The system consists of two parts, corresponding to the two departments.

The prototype translator translated these diagrams first to a CLKS, according to the semantics defined by Eshuis and Wieringa [6]: state names are translated to proposition symbols; actions become labels on the transitions.

Then, a second phase of the translator generated a Kronos input file. We have checked the following properties using Kronos:

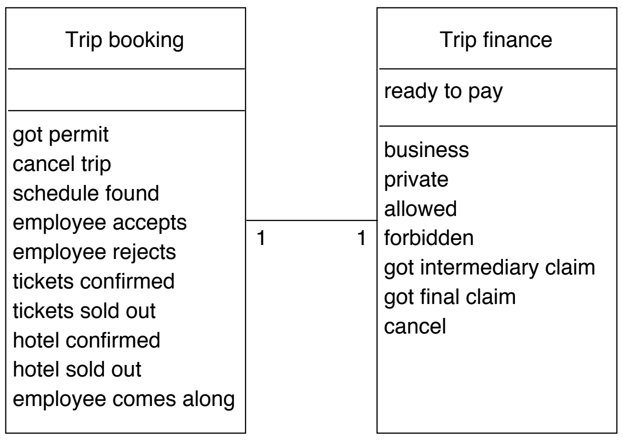

Figure 1: Lightweight class diagram for the ITO system.

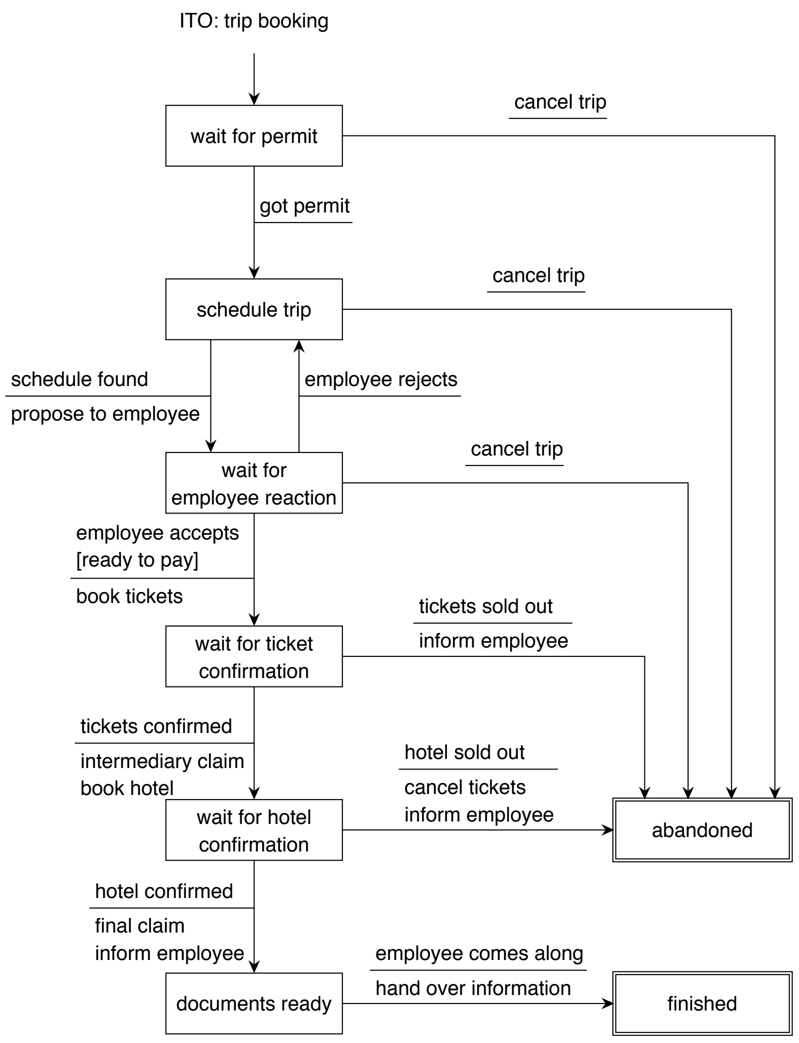

Figure 2: The ITO's process for a trip.

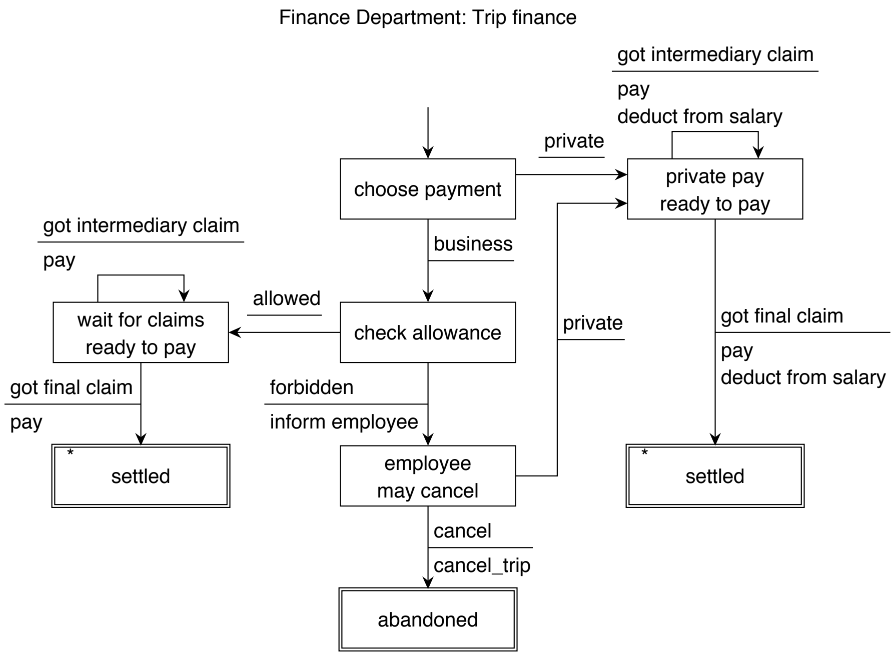

Figure 3: The financial department's process for a trip.

- "The system is non-Zeno" or, more exactly, "in every state, time may pass at least one second." (This technical property is needed because the model checker Kronos does not ensure it automatically.) This can be formalised in ATCTL as:

$$\mathsf{init} \to \forall\Box[\exists\top\ {}_{\mathbf{any}}\mathcal{U}^{\geq 1}\ \top]$$

- "No payment without allowance." We should state this more precise, saying "without allowance, all payments are deducted from the salary." This is formalised in ATCTL as:

$$\neg\exists(\top\ {}_{-\mathbf{allowed}}\mathcal{U}_{\mathbf{pay}\&-\mathbf{deduct}}\ \top)$$

- "No payment without statement of expenses."

$$\neg\exists(\top\ {}_{-\mathbf{gotclaim}}\mathcal{U}_{\mathbf{pay}}\ \top)$$

  Note the different formalisation of "payment" in this property and the one before.

Kronos reported that the property holds in all three cases.

2 Combining Statecharts And Probability

This work is still in progress. A first important result has been submitted as a joint article with Holger Hermanns and Joost-Pieter Katoen.

P-Statecharts: A simple extension of statecharts to describe probabilistic systems. We extend statecharts by P-pseudonodes Ⓟ which denote a non-trivial probability distribution. From the P-pseudonode, several arrows emanate, each with a probability and an action set. Further, the same drawing conventions as for statecharts hold.

Fig. 4 depicts a P-statechart which shows the behaviour of a *fair, but unreliable coin:* the event "flip" may or may not be ignored. If the system reacts, it outputs "heads" or "tails", each with 50 % chance. It is unspecified how (un)reliable the system is.

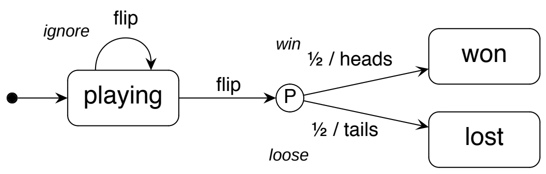

Figure 4: Example P-statechart.

2.1 Semantics

The extension of P-statecharts is independent from the semantics one uses. We have chosen, as a representative, the object-oriented semantics of Eshuis and Wieringa [6]. One could define a similar extension for other semantics, too.

Intuitive semantics for a single P-statechart. The intuitive behaviour of a P-statechart can be described as follows. The statechart is always in some state (which consists of one or several nodes). A P-edge is taken if the P-statechart is in the source node(s), the event of the edge happens and its guard holds. Then, the system chooses one of the possible results (probabilistically and nondeterministically); it leaves the source nodes, executes the chosen action and enters the chosen target nodes of the P-edge. More than one edge may be taken simultaneously.

Markov decision processes. The formal semantics of a collection of communicating P-statecharts is a Markov decision process (MDP).

An *MDP* is a quadruple $(S, \mathit{Distr}, L, s_0)$ where:

- S is a finite, non-empty set of states.
- Distr assigns to each state a finite, non-empty set of distributions on S.
- $L : S \to \mathbb{P}(AP)$ assigns to each state a set of atomic propositions.
- $s_0 \in S$ is the initial state.

In state s, the atomic propositions in $L(s)$ hold. Informally speaking, an MDP exhibits the following behaviour. Whenever the system is in state s, a probability distribution $\mu \in \mathit{Distr}(s)$ is chosen nondeterministically. Then, the system chooses probabilistically the next state according to the selected distribution μ.

To illustrate how P-statecharts are mapped onto MDPs, we consider the "unreliable coin" P-statechart from Fig. 4. The MDP semantics of this P-statechart is illustrated in Fig. 5. The inscriptions in circle-shaped states consist of a configuration (a set of nodes) and a set of input events (to which the system will react next). The inscriptions in rectangular states consist of a configuration and a set of edges (of which a maximal consistent subset is executed next). The names used for edges are shown in italics in Fig. 4.

2.2 Model Checking P-Statecharts

We express desired properties of a P-statechart using PBTL [2], a probabilistic branching time logic interpreted over MDPs. It allows one to express properties such as "the probability that a given system crashes within 13 steps without ever visiting certain states is at most 10^{-5}". In order to decide these properties, the non-determinism is resolved by means of schedulers.

Prism [9] is a model checker that checks whether an MDP satisfies some property expressed in PBTL. A typical output of the model checker is: "the probability in the initial state is actually $0, \ldots$, so the formula is true/false."

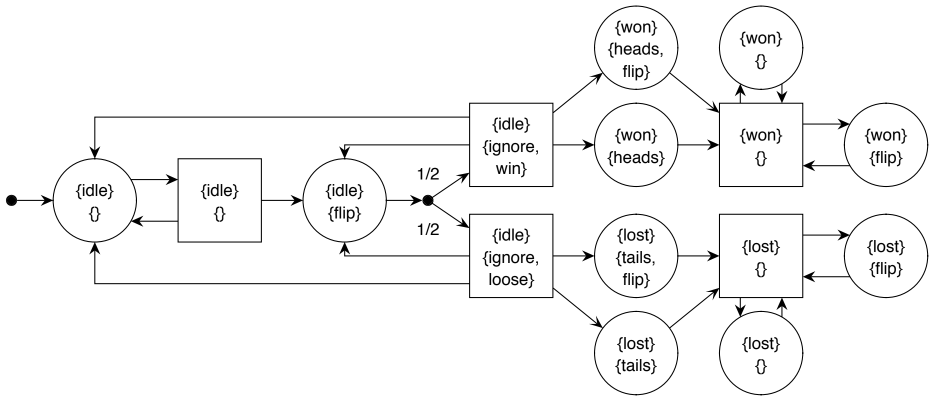

Figure 5: MDP semantics of the P-statechart of Fig. 4

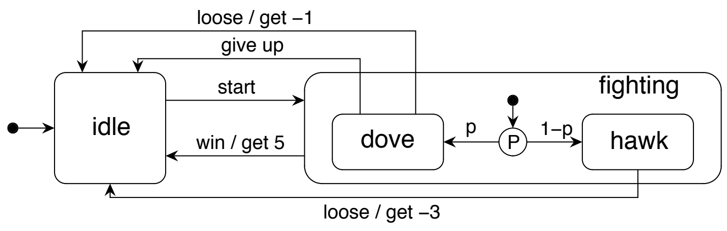

Figure 6: Statechart of a contestant in the hawk–dove-game.

2.3 Example: Hawk–Dove-Game

In theoretical biology, conflicts between animals are often analysed using simulation techniques. We consider the following variant of the hawk–dove-game [3, 10]. In a population of animals, individuals combat for some advantage (such as food, dominance, or mates), their success being measured in points. Individuals may fight using several strategies. In particular, we consider

Hawk strategy: Hawk-like individuals will fight with great effort, until they win the contest (+5 points) or are severely injured (−3 points).

Dove strategy: Dove-like individuals will fight with limited effort, until they win the contest (+5 points) or give up after some fight (−1 point). When facing a hawk, they immediately give up (±0 points).

We consider a small scenario with three individuals and an arbiter. In every round, the arbiter chooses nondeterministically a pair of individuals; they will be opponents in the next contest. The two individuals select probabilistically the hawk or dove strategy. The arbiter decides who wins. Fig. 6 and 7 show the P-statechart for one individual and the arbiter, respectively. The players all start off with 17 points and the individual scores may float in the interval $[0, 55]$ (otherwise they stop). Applying the MDP semantics together with some further optimisations (leaving out trivial intermediary states, encoding the configuration efficiently) leads to a system of 3,147,947 reachable states. The size of the state space is mainly dominated by the integer variables storing the scores. Different scenarios were checked with the model checker Prism [9] where each scenario consisted of different types of animals. These types were generated by taking different values for p, the probability to behave like a dove. Formulas are checked for the initial state. The three considered scenarios are the following.

One daring and two careful players. This is a scenario with two individuals (c_1 and c_2) for which $p = 0.75$ and one individual (d) with $p = 0.5$. The probability that any individual dies (its

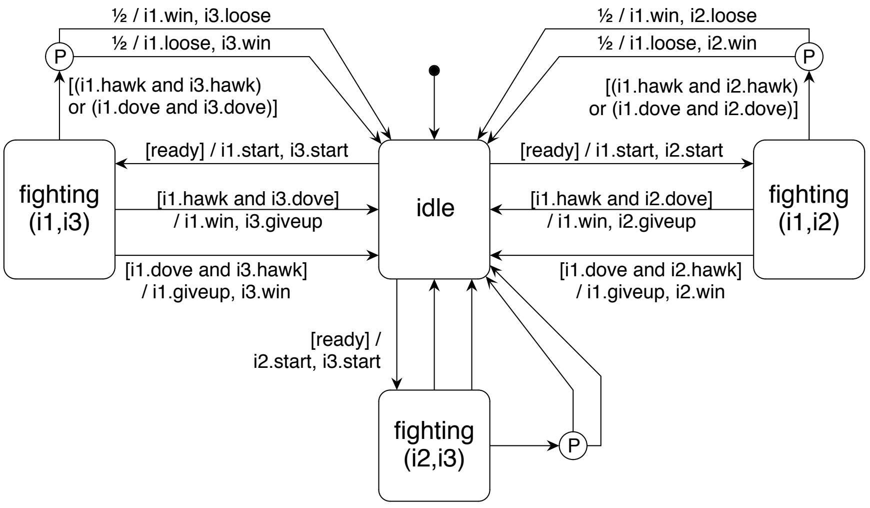
½ / i1.win, i3.loose
½ / i1.loose, i3.win
P
[(i1.hawk and i3.hawk) or (i1.dove and i3.dove)]
½ / i1.win, i2.loose
½ / i1.loose, i2.win
P
[(i1.hawk and i2.hawk) or (i1.dove and i2.dove)]
[ready] / i1.start, i3.start
[ready] / i1.start, i2.start
fighting (i1,i3)
idle
fighting (i1,i2)
[i1.hawk and i3.dove] / i1.win, i3.giveup
[i1.hawk and i2.dove] / i1.win, i2.giveup
[i1.dove and i3.hawk] / i1.giveup, i3.win
[i1.dove and i2.hawk] / i1.giveup, i2.win
[ready] / i2.start, i3.start
fighting (i2,i3)
P

Figure 7: Statechart of the arbiter in the hawk–dove-game. We have omitted some edge labels from and to node fighting(i2,i3), which are analogous the other fighting nodes.

points drop below 0) turns out to be very small, with the daring individual running a higher risk of being killed, since

$$\mathcal{P}_{\leq 10^{-7}}[(\neg dead(c_1) \wedge \neg dead(d))\,\mathcal{U}\,dead(c_2)]$$

is refuted (and likewise with c_1 and c_2 reversed), but

$$\mathcal{P}_{\leq 10^{-7}}[(\neg dead(c_1) \wedge \neg dead(c_2))\,\mathcal{U}\,dead(d)]$$

holds. The actual probability of d dying first is at most (depending on the scheduler) $7.206 \cdot 10^{-7}$, while the probability of the careful one dying first is at most $5.923 \cdot 10^{-8}$ (each). On the other hand, the daring individual is likely to outperform the others on accumulating a certain number of points, say 37. This follows from verifying:

$$\mathcal{P}_{<0.5}[(points_{c_1} < 37 \wedge points_d < 37)\,\mathcal{U}\,points_{c_2} \geq 37] \text{ which is valid, and}$$

$$\mathcal{P}_{\leq 0.75}[(points_{c_1} < 37 \wedge points_{c_2} < 37)\,\mathcal{U}\,points_d \geq 37] \text{ which is invalid.}$$

Three aggressive players. In this scenario each animal (d_1, d_2, d_3) plays hawk with probability 0.9 (i.e., $p = 0.1$). The probability that some of the individuals dies is relatively high, e.g.,

$$\mathcal{P}_{\leq 0.01}[(\neg dead(d_1) \wedge \neg dead(d_2))\,\mathcal{U}\,dead(d_3)]$$

is refuted (and likewise for the permutations of the d_i). So, there are schedulers which will lead to d_3 dying first with more than 1 % chance. The probability that one of the individuals gets more than 37 points within 100 steps is always less than 0.75, as

$$\mathcal{P}_{<0.75}[\Diamond^{\leq 100}\ (points_{d_1} \geq 37)] \text{ holds.}$$

Three careful players. In the opposite situation (the three individuals play dove with probability 0.9), the individuals (c_1, c_2, c_3) are less likely to die and more likely to get a reward fast. The probability that any of the individuals dies is rather low as, e.g.,

$$\mathcal{P}_{\leq 10^{-10}}[(\neg dead(c_1) \wedge \neg dead(c_2))\,\mathcal{U}\,dead(c_3)] \text{ holds.}$$

So, for any scheduler, the probability of c_3 dying first never exceeds 10^{-10}. The probability that one of the individuals gets more than 37 points within 100 steps turns out to be greater than 0.8, since

$$\mathcal{P}_{\leq 0.8}[\Diamond^{\leq 100}\ (points_{c_1} \geq 37)] \text{ is refuted.}$$

Conclusion. As a general conclusion of the experiments we may state that it is good for a population as a whole if the animals are careful; but an individual may be at an advantage if it is more daring than the others.

3 A Language For Stochastic Time

We plan to look at similar extensions of statecharts for stochastic models. In a stochastic model, the duration of certain actions or time intervals is distributed according to a probability distribution. A simple example is radioactive decay: the time between "now" and the moment that a nucleus decays is distributed according to the exponential probability distribution. We want to give a language similar to P-statecharts, which is easy to use, for stochastic systems.

There are also model checkers for stochastic systems. So, we want to use these model checkers in a similar way to check properties of systems described by our language.

All this work is intended to spread probabilistic and stochastic model checking further. These newer variants of model checking have been developed the past years and should now be presented in a form which is similar to known techniques, to make it easier to apply them in real cases.

References

[1] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *Information and computation*, 104:2–34, 1993.

[2] Christel Baier and Marta Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11:125–155, 1998.

[3] Philip H. Crowley. Hawks, doves, and mixed-symmetry games. *Journal of Theoretical Biology*, 204(4):543–563, June 21 2000.

[4] Rocco De Nicola and Frits Vaandrager. Action versus state based logics for transition systems. In I. Guessarian, editor, *Semantics of systems of concurrent processes: ... Proceedings*, volume 469 of *LNCS*, pages 407–419, Berlin, 1990. Springer.

[5] E. Allen Emerson and Edmund M. Clarke. Using branching time temporal logic to synthezise synchronisation skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[6] Rik Eshuis and Roel Wieringa. Requirements-level semantics for UML statecharts. In Scott F. Smith and Carolyn L. Talcott, editors, *Formal Methods for Open Object-Based Distributed Systems IV : ... FMOODS*, pages 121–140, Boston, 2000. Kluwer Academic Publishers.

[7] David N. Jansen and Roel Wieringa. Extending ctl with actions and real-time. In *International Conference on Temporal Logic*, Leipzig, 2000.

[8] David N. Jansen and Roel Wieringa. Reducing the extensions of CTL with actions and real time. Technical report, Universiteit Twente, Enschede, December 2000. TR-CTIT-00-27.

[9] M. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic symbolic model checker. In Joost-Pieter Katoen and P. Stevens, editors, *[Proceedings of TACAS 2002]*, volume 2280 of *LNCS*, pages 51–65, Berlin, 2002. Springer.

[10] J. Maynard Smith and G. R. Price. The logic of animal conflict. *Nature*, 246, November 1973.

[11] H. W. M. Verbeek, T. Basten, and W. M. P. van der Aalst. Diagnosing workflow processes using Woflan. Technical Report 99/02, Technische Universiteit, Eindhoven, 1999.

[12] Roel Wieringa and Jan Broersen. A minimal transition system semantics for lightweight class- and behavior diagrams. In *ICSE98 Workshop on Precise Semantics for Software Modeling Techniques*, pages 129–151, München, 1998. Technische Universität. Report TUM-I9803.

[13] Sergio Yovine. Kronos: A verification tool for real-time systems. *Springer international journal of software tools for technology transfer*, 1(1/2), 1997.