

GAMES 2005

Paris, September 21 – 24, 2005

Games and Logics over Dynamically Changing Structures

Philipp Rohde

Informatik VII

RWTHAACHEN

Sabotage Game and Sabotage Modal Logic

Classical logics: underlying structures are assumed to be static.

Investigation: Logics over dynamically changing structures.

Sabotage proposed by van Benthem / see GAMES '03: Focus on deletion of objects.

Applications: Server networks where connections break down. Navigation systems: cope with roadworks and traffic jams. Representation of knowledge: increase in knowledge \sim removal of uncertainty relations. Euler's seven bridges of Königsberg.

Sabotage game: One player moves along edges within network; other player removes edges. Reachability of vertices as winning condition.

Finite game! Algorithmic complexity: Solving the game is PSPACE-complete.

Sabotage modal logic SML: Modal logic ML + transition deletion modality.

Semantics for Kripke structure \mathcal{K} , state s .

Standard modality:

$(\mathcal{K}, s) \models \diamond_a \varphi$ iff there is transition $s \xrightarrow{a} s'$ of \mathcal{K} such that $(\mathcal{K}, s') \models \varphi$

Sabotage modality:

$(\mathcal{K}, s) \models \diamond_a \varphi$ iff there is transition $t \xrightarrow{a} t'$ of \mathcal{K} such that $(\mathcal{K} \setminus \{t \xrightarrow{a} t'\}, s) \models \varphi$

Polynomial embedding into FO.

SML lacks tree model property, lacks finite model property, is not invariant under bisimulation.

Logic	Combined MC	Formula	Program	Satisfiability
ML	PTIME-complete	in PTIME	in PTIME	PSPACE-complete
SML	PSPACE-complete	in PTIME	in PTIME	undecidable
FO	PSPACE-complete	PSPACE-complete	in PTIME	undecidable

Sabotage makes modal logic much harder; SML resembles FO.

Asymmetry: Local movements (standard modalities) vs. global deletion (sabotage modalities).

But: Same picture when also deletion is subject to a locality condition (adjacent sabotage / path sabotage): Local deletion does not lower complexities!

Results in MFCS '03, FSTTCS '03 (joint work with Ch. Löding), and CSL '04.

Main limitation of ML / SML: Lack of mechanism for unbounded iteration or recursion.

Add constructors for forming fixed-points of relational operators to logic.

Recall: ML + least / greatest monadic fixed-points \rightsquigarrow μ -calculus L_μ .

Analogously: **Sabotage μ -calculus SL_μ** .

In negation normal form; variables occur positively.

Formula φ , Kripke structure \mathcal{K} with state set S , valuation $\mathcal{V} : \text{Var}(\varphi) \rightarrow 2^S$ of variables:

$\|\varphi\|_{\mathcal{V}}^{\mathcal{K}}$ gives set of states in which φ is true.

X free variable of φ :

$$\|\mu X.\varphi\|_{\mathcal{V}}^{\mathcal{K}} := \text{lfp} (A \mapsto \|\varphi\|_{\mathcal{V}[X:=A]}^{\mathcal{K}}).$$

Monadic operator $A \mapsto \|\varphi\|_{\mathcal{V}[X:=A]}^{\mathcal{K}}$ monotone: Least fixed-point exists by Knaster-Tarski and

$$\|\mu X.\varphi\|_{\mathcal{V}}^{\mathcal{K}} = \bigcap \{A \subseteq S \mid \|\varphi\|_{\mathcal{V}[X:=A]}^{\mathcal{K}} \subseteq A\}.$$

Analogous for $\nu X.\varphi$ / greatest fixed-point.

Remark: Fundamental difference between \diamond_a and \heartsuit_a with respect to (monadic) fixed-points.

Inductive fixed-point construction: Movements are passed to the next stage; deletion of transitions is 'encapsulated' within a stage. For every inductive step, deletion is restored to the situation at the beginning.

Examples

Example 1: $\varphi = \nu X. \mu Y. (\Box \Diamond X \vee \Diamond Y)$

φ expresses that the unraveling of a structure contains a perfect subtree (each path contains infinitely many splitting points).

Example 2: SML-formulae can ensure that origin has infinitely many successors (infinite *width* of models). With fixed-points, we can enforce an infinite *depth* of models:

$$\begin{aligned}\psi &:= \Diamond_b \top \wedge \Diamond_b (\Box_b \perp \wedge \nu Y. (\Box_a (\Diamond_b \top \wedge Y))) \\ \varphi &:= \nu X. (\Diamond_a X \wedge \psi).\end{aligned}$$

$(\mathcal{K}, t) \models \psi \implies t$ has exactly one b -transition and if it is removed, then every state reachable from t by a non-empty a -path still has b -successors.

In particular: Every state reachable from t by a non-empty a -path is distinct from t .

$(\mathcal{K}, s) \models \varphi \implies$ there is an infinite path $\pi = s_0 \xrightarrow{a} s_1 \xrightarrow{a} s_2 \xrightarrow{a} \dots$ with $s_0 = s$ and $(\mathcal{K}, s_i) \models \psi$ for every $i \in \mathbb{N}$.

Thus: π consists of pairwise distinct elements, i.e., π is a simple path of infinite length.

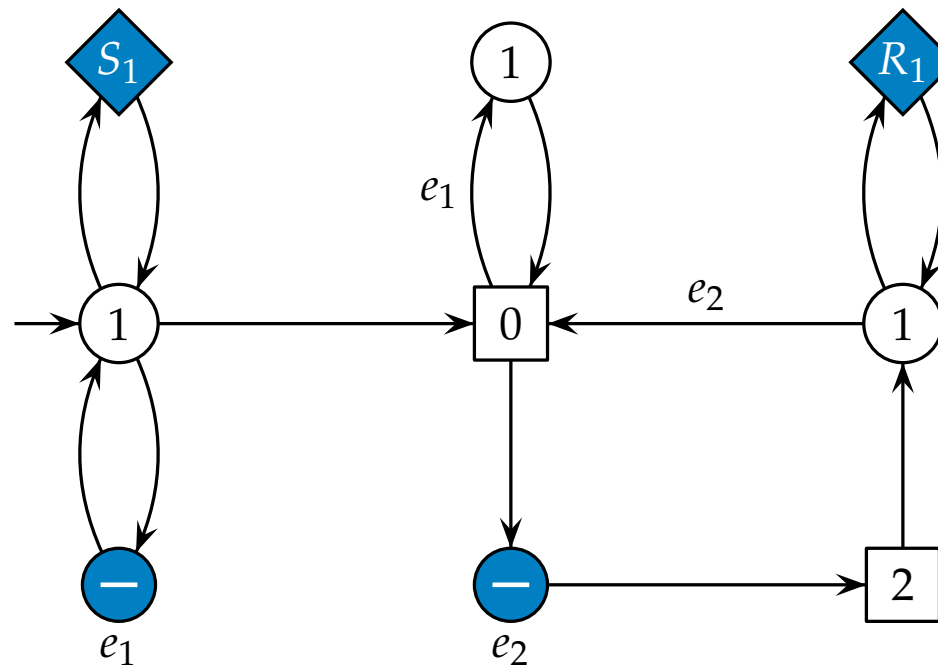
- Model checking for SL_μ is PSPACE-complete.
(SML fragment of SL_μ / embedding of SL_μ into monadic LFP)
- Program complexity: Model checking for SL_μ with fixed formula is PTIME-complete.
(μ -calculus fragment of SL_μ / embedding into LFP)
- SL_μ lacks finite model property, is not invariant under bisimulation.
- Satisfiability, Finite Satisfiability, and Infinity Axiom for SL_μ are undecidable.

Question: Model checking game for SL_μ (analogous to parity games for L_μ) ?

Backup parity game: Extension of sabotage game and standard parity game.

Ingredients: Fixed number of registers. Type of vertices:

- Player-based choice of direction of movement (as with parity games)
- Deletion of edges
- Storing current appearance of arena into register
- Restoring arena out of register



- Player-based choices / priorities only for movement vertices; exactly one non-deletable outgoing edge of other vertices
- Vertex is sink / deletion not possible: respective player loses
- Infinite play: winner according to parity condition (highest priority counts)

Complexity issue: Access to registers follows stack discipline:

- Storing of values overwrites all lower registers
- Restoring out of register erases values of all lower registers

Backup Parity Games

- Positional determinacy (position comprises current appearance / values of registers!)
- Automaton strategies suffice
- Family of games without registers, size $\mathcal{O}(k)$: automata need 2^k states
- Family of games with m registers, size $\mathcal{O}(m^3)$: automata need $m!$ states

Theorem: Fixed number of registers / stack discipline: Solving the game is PSPACE-complete.

Thus: Parity condition + storing / restoring with stack discipline does not make the (finite) sabotage game algorithmically harder.

Theorem: Without stack discipline: EXPTIME-complete, even for three registers.

Theorem: Backup parity games serve as model checking games for sabotage μ -calculus (size of game quadratic in $|\mathcal{K}| \cdot |\varphi|$).

Model checking for SL_μ via backup games not optimal yet:

Solving the game is known to be PSPACE only for fixed number of registers. But number of registers corresponds to depth of nested fixed-point operators of SL_μ -formula.

Open questions:

- Can backup games be solved in PSPACE, regardless of number of registers?
PSPACE-subclass corresponding to model checking games?
- Can we express the winning condition by SL_μ -formulae (in polynomial size)?
- Extensions of sabotage modal logic where movements *and* deletions can be iterated, e.g., reachability while deleting edges?

Thank you for your attention!