Games for Logics with Partial Order Models

Julian Gutierrez and Julian Bradfield

Laboratory for Foundations of Computer Science
School of Informatics, University of Edinburgh

GaLoP, March, 2009

Motivation

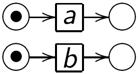
Avoid the use of interleaving models of concurrency.

But why?

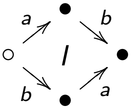
1. Model-checking I: suffer from state explosion problem.
2. Model-checking II: use of partial order reduction methods.
3. Model-checking III: verification beyond temporal properties.
4. Equivalence-checking: verification of infinite state systems.
5. Synthesis: produce "global" components (automata).
6. Analysis: local reasoning on parallel components.
7. Game semantics: perhaps not the right models for logics with an explicit notion of concurrency or independence.

Interleaving and Partial Order Models of Concurrency

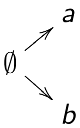
$$a.b + b.a \quad \equiv_{intm} \quad a \parallel b \quad \equiv_{pom} \quad a \; I \; b$$



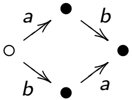
(a) Petri Nets



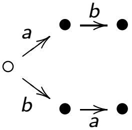
(b) TSI



(c) Event Structures



(d) LTS



(e) ST

Partial Order Models of Concurrency: Features

Behaviour:

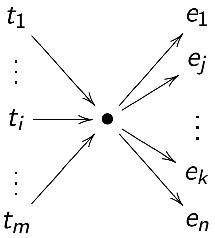
1. Concurrency: parallel computations.
2. Causality: sequential computation.
3. Conflict: deterministic and nondeterministic choices.

Structure:

1. A set of states S.
2. A set of events or transitions T.
3. An independence relation I on elements of T.
4. An alphabet of labels Σ (only for labelled models).

Local Dualities in Partial Order Models of Concurrency

A new approach to observing concurrent behaviour introduced in [Gut09] - FoSSaCS'09.



1. First duality: concurrency vs. causality.

$$\begin{array}{rcll} t_i \mathrel{I} e_j & \Leftrightarrow & t_i \ominus e_j & \text{concurrent, but linearized.} \\ \neg(t_i \mathrel{I} e_j) & \Leftrightarrow & t_i \leq e_j & \text{causally dependent.} \end{array}$$

2. Second duality: concurrency vs. conflict.

$$\begin{array}{rcll} e_j \mathrel{I} e_k & \Leftrightarrow & e_j \otimes e_k & \text{immediately concurrent (same trace).} \\ \neg(e_j \mathrel{I} e_k) & \Leftrightarrow & e_j \# e_k & \text{in conflict (different traces).} \end{array}$$

Traces and Sets of Transitions

Support sets:

- Maximal Set: $P_{max}(s) = \{t \in T \mid src(t) = s\}$ (all traces).
- Conflict-free set: $E \subseteq P_{max}(s)$ s.t. $\forall t_1, t_2 \in E.\ t_1 \neq t_2 \Rightarrow t_1 \otimes t_2$ (one single trace).

Notation:

$$E \sqsubseteq R \stackrel{\text{def}}{=} E \subseteq R, \text{ s.t. } E \text{ is a conflict-free set, i.e., a trace.}$$

$$P_1 \uplus P_2 \stackrel{\text{def}}{=} P_1 \cup P_2, \text{ s.t. } P_1 \cap P_2 = \emptyset \wedge P_1 \neq \emptyset \wedge P_2 \neq \emptyset$$

Separation Fixpoint Logic (SFL)

SFL is an extension of the Modal Mu-Calculus that can express properties of partial order models of concurrency.

Syntax: SFL has formulae ϕ built from a set Var of variables $\mathrm{Y}, \mathrm{Z}, \ldots$ and a set $\mathcal{L}$ of labels $a, b, \ldots$ by the following grammar:

ϕ	$::=$	Z	Variables
		$\mid \neg\phi_1 \mid \phi_1 \wedge \phi_2$	Boolean operators
		$\mid \langle K \rangle_c \phi_1 \mid \langle K \rangle_{nc} \phi_1$	Modal operators (Duality conc. vs. caus.)
		$\mid \phi_1 * \phi_2$	Structural operator (Duality conc. vs. conf.)
		$\mid \mu Z.\phi_1$	Fixpoint operator

SFL: Derived Operators and Notation

1. Derived Operators:
   - $\phi_1 \vee \phi_2 \stackrel{\text{def}}{=} \neg(\neg\phi_1 \wedge \neg\phi_2)$
   - $\phi_1 \bowtie \phi_2 \stackrel{\text{def}}{=} \neg(\neg\phi_1 * \neg\phi_2)$
   - $[K]_c\, \phi_1 \stackrel{\text{def}}{=} \neg\langle K\rangle_c \neg\phi_1$
   - $[K]_{nc}\, \phi_1 \stackrel{\text{def}}{=} \neg\langle K\rangle_{nc} \neg\phi_1$
   - $\nu Z.\phi_1 \stackrel{\text{def}}{=} \neg\mu Z.\neg\phi_1\, [\neg Z/Z]$
2. Abbreviations:
   - $\mathrm{ff} \stackrel{\text{def}}{=} \mu Z.Z$
   - $\mathrm{tt} \stackrel{\text{def}}{=} \nu Z.Z$
   - $[-]\,\phi \stackrel{\text{def}}{=} [\mathcal{L}]\,\phi$
3. Formulae in Positive Normal Form.

A Concrete Model

A *TSI-based SFL model* $\mathfrak{M}$ is a TSI $\mathfrak{T} = (S, T, \Sigma, I)$ together with a valuation $\mathcal{V} : \mathrm{Var} \to 2^{\mathfrak{S}}$, where:

- $\mathfrak{S} = S \times \mathfrak{P} \times \mathfrak{A}$ is the set of tuples (s, P, t_a) s.t.:
- $s \in S$ is a state of the TSI,
- $P \in \mathfrak{P}$ is a support set at s,
- $t_a \in \mathfrak{A} = T \cup \{t_\epsilon\}$ is a transition, and
- a is an action label in $\Sigma \cup \{\epsilon\}$.

Remarks:

- A tuple (s, P, t_a) of a model $\mathfrak{M}$ is called a *process*.
- The initial process of the system is the tuple $H = (s_0, P_{max}, t_\epsilon)$.

SFL Sublogics

Logic	Synt. rest.	$\ominus$ vs. $\leq$	$\otimes$ vs. $\#$
$L\mu$	plain modalities only/$*$-free	NO	NO
$CL\mu$	$*$-free	YES	NO
$SL\mu$	plain modalities only	NO	YES
SFL	none	YES	YES

Encoding plain modalities:

$$\begin{array}{rcl} \langle K \rangle \phi & = & \langle K \rangle_c \phi \vee \langle K \rangle_{nc} \phi \\ [K]\,\phi & = & [K]_c\,\phi \wedge [K]_{nc}\,\phi \end{array}$$

Trace MSO Model Checking Games

Main idea behind the game: a player can see independence and therefore can play traces, i.e., sets of independent transitions. This is reflected in the rules and winning conditions of the game.

1. Players: Adam (Falsifier) and Eve (Verifier).
2. Board: A set of configurations in $\mathfrak{B} = \mathfrak{S} \times \mathit{Sub}(\phi)$.
3. Rules: next slide...
4. Winning conditions: In finite plays a player wins if the other cannot make a move. In infinite plays the winner depends on the fixpoints.

A play is NOT alternating. The player to make a move is defined by $\mathit{Sub}(\phi)$.

Trace MSO Model Checking Games: Rules

FIXPOINT OPERATORS					
(FP)	$\dfrac{H \vdash \sigma Z.\phi}{H \vdash Z}$	$\sigma \in \{\mu, \nu\}$	(VAR)	$\dfrac{H \vdash Z}{H \vdash \phi}$	$fp(Z) = \sigma Z.\phi$
BOOLEAN OPERATORS					
$(\vee)$	$\dfrac{H \vdash \phi_0 \vee \phi_1}{H \vdash \phi_i}$	$\exists i : \quad i \in \{0, 1\}$	$(\wedge)$	$\dfrac{H \vdash \phi_0 \wedge \phi_1}{H \vdash \phi_i}$	$\forall i : \quad i \in \{0, 1\}$

MODAL OPERATORS

$$(\langle\,\rangle_c) \quad \frac{(s, R, t_a) \vdash \langle K \rangle_c \phi}{(s', R'_{max}, t_b) \vdash \phi} \quad \exists b : \quad b \in K, s \xrightarrow{b} s' = t_b \in R, t_a \leq t_b$$

$$(\langle\,\rangle_{nc}) \quad \frac{(s, R, t_a) \vdash \langle K \rangle_{nc} \phi}{(s', R'_{max}, t_b) \vdash \phi} \quad \exists b : \quad b \in K, s \xrightarrow{b} s' = t_b \in R, t_a \ominus t_b$$

$$([\,]_c) \quad \frac{(s, R, t_a) \vdash [K]_c\, \phi}{(s', R'_{max}, t_b) \vdash \phi} \quad \forall b : \quad b \in K, s \xrightarrow{b} s' = t_b \in R, t_a \leq t_b$$

$$([\,]_{nc}) \quad \frac{(s, R, t_a) \vdash [K]_{nc}\, \phi}{(s', R'_{max}, t_b) \vdash \phi} \quad \forall b : \quad b \in K, s \xrightarrow{b} s' = t_b \in R, t_a \ominus t_b$$

STRUCTURAL OPERATORS

$$(*) \quad \frac{(s, R, t) \vdash \phi_0 * \phi_1}{(s, R_i, t) \vdash \phi_i} \quad \exists f, \forall i : \quad f \in \mathfrak{P}^{\{0,1\}}, R_i \uplus R_{1-i} \sqsubseteq R, i \in \{0, 1\}$$

$$(\bowtie) \quad \frac{(s, R, t) \vdash \phi_0 \bowtie \phi_1}{(s, R_i, t) \vdash \phi_i} \quad \forall f, \exists i : \quad f \in \mathfrak{P}^{\{0,1\}}, R_i \uplus R_{1-i} \sqsubseteq R, i \in \{0, 1\}$$

Trace MSO Model Checking Games: Properties

- Closed under dual games.
- Eve preserves falsity and can preserve truth with her choices.
- Adam preserves truth and can preserve falsity with his choices.
- In any infinite play there is a unique syntactically outermost variable that occurs infinitely often.
- In infinite plays rule (VAR) must be applied infinitely often (important: infinite state systems with finite-branching).
- Winning conditions ensure a unique winner.

Main Results

Theorem: Trace MSO Model-Checking Games are Sound.

Theorem: Trace MSO Model-Checking Games are Complete.

Corollary: Trace MSO Model Checking Games are Determined.

Other Results

1. The winning strategies in the Trace MSO model-checking game of Separation Fixpoint Logic (SFL) are history-free.
2. In interleaving models of concurrency, the Trace MSO Model-checking games for SFL coincide with Stirling's Local Model-checking games for the Modal Mu-Calculus.

A Hintikka Game Semantics for SFL

$H \models^{\mathfrak{T}}_{\mathcal{V}} \phi$ iff Eve has a history-free winning strategy in the Trace MSO model-checking game $\mathcal{G}_{\mathfrak{M}}(H, \phi)$

1. This game model does not make use of the one-step interleaving semantics of the partial order model being considered.
2. Since Trace MSO Model-Checking Games are determined, this Game Theoretic Semantics (*à la* Hintikka) is, as well as the denotational one (*à la* Tarski), compositional.

Beyond Temporal Properties: Multi-Agent Systems

Agents:

- Γ is a finite set of agents, and
- $\mathcal{A} : T \to \Gamma$ is a mapping from transitions to agents.

Consistency of global actions:

- if $t_1 \sim t_2$ then $\mathcal{A}(t_1) = \mathcal{A}(t_2)$.

A distributed system:

- if $\mathcal{A}(t_1) \neq \mathcal{A}(t_2)$ then $lbl(t_1) \neq lbl(t_2)$.

Consistency of formulae:

- $\langle a \rangle^\alpha \phi$ is *well-defined* iff $a = lbl(t)$ and $\alpha = \mathcal{A}(t)$ for some $t \in T$ and $\alpha \in \Gamma$.
- $\langle K \rangle^\alpha \phi = \bigvee_{a \in K} \langle a \rangle^\alpha \phi$.

Multi-Agent Systems - Example

$$\psi = [-]^{\beta} \langle - \rangle^{\alpha}_{nc} \mu Z.\phi \vee \langle - \rangle^{\alpha}_{c} Z$$

Formula ψ expresses that there is an agent α (the system) that can satisfy ϕ regardless the behaviour of an adversarial agent β (the environment). Informally, ψ says "whatever you (the environment) do, I (the system) can get to ϕ, though I may first have to do some things that do not depend on what you did."

Conclusions and Current/Future Work

1. The approach to defining games presented here, i.e., players allowed to play sets of elements, can help define:
   - Sound and complete, and therefore determined, games in partial order models of concurrency.
   - compositional game semantics for logics of concurrency.
2. The games presented here naturally capture the behaviour of partial order models, since the one-step interleaving semantics of those systems need not be considered.
3. Trace MSO Model-checking games deal equally well with both interleaving and partial order models of concurrency.
4. Temporal verification of regular but infinite partial order models.
5. Synthesis of asynchronous circuits.