

Game Semantics Supported Component Verification

Aleksandar Dimovski Ranko Lazić

Department of Computer Science
University of Warwick, UK

BCTCS, 2006

Traditional approaches for software model checking

- Operational semantics:
Program denotes a state transition system
- Finite models by predicate abstraction
- Tools:
SLAM [Microsoft], BLAST [Berkeley], Magic [CMU]

Objectives, motivations

Game semantics yields algorithms for software model checking

- based on Regular languages for 2nd-order finitary IA
[Abramsky, Ghica, Murawski and Ong, TACAS 2004]
- based on Visibly pushdown automata for 3rd-order finitary IA
[Murawski, Walukiewicz and Ong, FoSSaCS 2005]

Advantages of Game semantics based approach for software model checking:

- model for any open program fragment with higher-order procedures
- fully abstract semantic model
(i.e. the model is sound and complete)
- compositional as denotational semantics
- generation of minimal models

Compositional verification

- addresses state explosion problem
- “divide and conquer” approach
- assume–guarantee reasoning

- $\llbracket \Gamma \vdash \text{let } f_1 \text{ be } N_1 \text{ in}$
 $\quad \text{let } f_2 \text{ be } N_2 \text{ in}$
 $\quad \dots$
 $\quad \text{let } f_k \text{ be } N_k \text{ in}$
 $\quad \quad M : T' \rrbracket =$

$$\llbracket \Gamma \vdash N_1 \rrbracket \wp \left(\llbracket \Gamma, f_1 \vdash N_2 \rrbracket \wp \left(\dots \wp$$

$$\llbracket \Gamma, f_1, \dots, f_{k-1} \vdash N_k \rrbracket \wp \llbracket \Gamma, f_1, \dots, f_k \vdash M : T' \rrbracket \right)$$

Game semantics: the idea

- Computation seen as a play between a program – Player P and its environment (context) – Opponent O
- A play consists of a sequence of moves (questions and answers), alternating between players
- Model of a program is a *strategy* (set of plays) for Player P to respond to moves of Opponent O

- Types are interpreted as *games*
- A game G is
 - a set of moves M_G
 - a labelling function $\lambda_G : M_G \rightarrow \{O, P\} \times \{Q, A\}$
 - an enabling relation, \vdash_G
 - legal plays, $P_G \subseteq M_G^*$

- Programs are interpreted as *strategies*
- A strategy σ for a game G is an even-length prefix-closed non empty set of legal plays of G
- Composition = “CSP-style parallel composition plus hiding”

Abstracted Idealized Algol (AIA)

- Imperative features, locally-scoped variables, call-by-name procedures
- Integer abstractions:
 $[] = \{\mathbb{Z}\}$
 $[n, m] = \{ \langle n, n, n + 1, \dots, m - 1, m, \rangle m \}, \text{ for } n \leq 0 \leq m$
- *abort* command causes abnormal termination
- Operational semantics:
 $1 +_{[0,1]} 1 \rightarrow \{2, 3, 4 \dots\} \rightarrow > 1$
- For AIA_2 , game-semantic models are regular languages

Assume-Guarantee proof rule

$$\frac{\begin{array}{l} \llbracket \Gamma, f : T \vdash M : T' \rrbracket \text{ ; } \sigma \text{ is SAFE} \\ \llbracket \Gamma \vdash N : T \rrbracket^\dagger \leq \sigma \end{array}}{\llbracket \Gamma \vdash \text{let } f \text{ be } N \text{ in } M : T' \rrbracket \text{ is SAFE}}$$

- Given $\llbracket \Gamma, f : T \vdash M : T' \rrbracket$, define a *weakest assumption strategy* σ_W , which contains legal plays from $P_{\llbracket T \rrbracket}$ which, when simulated on $\llbracket \Gamma, f : T \vdash M : T' \rrbracket$ do not produce any unsafe plays
- For σ_W , AG proof rule is guaranteed to return a conclusive result
- Angluin's L^* algorithm is used for learning σ_W

L^* learns a strategy σ for G via queries

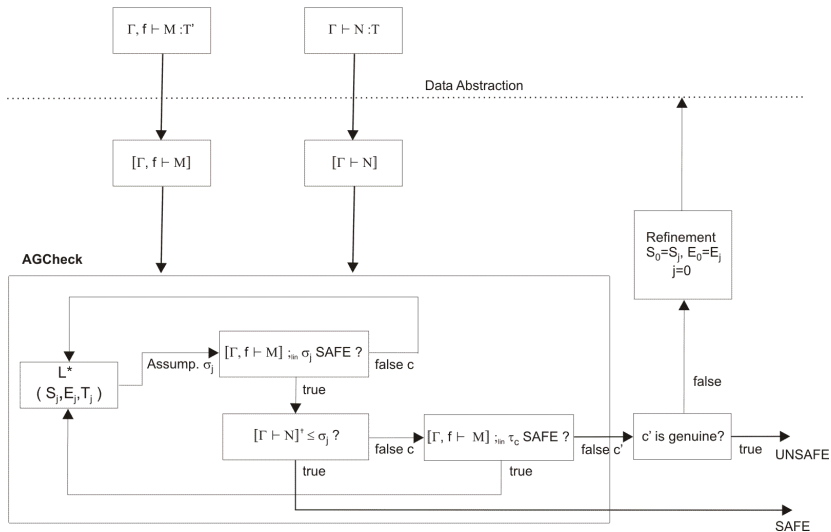
- *Membership query*: given $s \in (O_G P_G)^*$, whether $s \in \sigma$?
- *Equivalence query*: given a DFA D , whether $\mathcal{L}(D) = \sigma$?

L^* will generate the unknown strategy using at most $n - 1$ equivalence queries, where n is the number of states in the minimal DFA

Queries are implemented using model checking

- *Membership query*: given s , build a strategy $\sigma_s = \{s' \mid s' \sqsubseteq^{\text{even}} s\}$; then model check $\llbracket \Gamma, f \vdash M \rrbracket ; \sigma_s$ for safety
- *Equivalence query*: given a DFA D , model check two premises of the AG rule

Verification procedure



- Concurrency, 3rd-order, call by value, pointers, ...
- Predicate abstraction