Pulse: Plural To EVMDD-SMC
The Compiler and Model Generator

Ijaz Ahmed
Néstor Cataño
Radu I. Siminiceanu

University of Madeira, Portugal
National Institute of Aerospace, Virginia, USA

August 2, 2011
Plural and EVMDD-SMC

Pulse

Contributions

Abstract Models of Specifications
States Space and Relations

Translation Algorithm
Translation Algorithm (Co.)

Model Structure
CTL Properties

Demonstration through Eclipse

Conclusion
Plural and EVMDD-SMC

▶ Plural
A lightweight verification tool of Java programs by CMU
Verify access permissions and typestates
An Eclipse plug-in based on static analysis

▶ EVMDD-SMC
A symbolic model checking tool by NIA
Orders of magnitude faster than SAL model checker
Less syntactic sugar and edge value decision diagrams EVMDD
Pulse: Plural To EVMDD-SMC

- Translates Plural specification into EVMDD-SMC model
- Depends on PluralAnnotationsAnalysis
- Uses the Antlr parser generator
- An Eclipse plug-in like Plural
Contributions

- Absence of sink (deadlocked) states
- Typestate transition matrix (Plaid!)
- Possible ways to concurrency (Plaid!)
- Correct use of the access permission
Abstract Models of Specifications

- Access Permissions associated to object reference $r_i^j$: 
  $ap_i^j \in AP = \{\bot, \text{Unique, Full, Pure, Immutable, Share}\}$

- Typestate associated to object reference $r_i^j$: 
  $ts_i \in TS_i = \{\bot\} \cup \{t_{i1}^1, \ldots, t_{ih_i}^i\}$

- Program Counter associated to method $m_i$: 
  $(pc_i^j) \in PC_i = \{\text{Exe, notExe}\} \times (\{\bot\} \cup \{M_{i1}^1, \ldots M_{im_i}^{mi}\})$
States Space and Relations

- Set of potential global states $S$:

$$S = \prod_{i=1}^{c} \left( \{ \bot, t_1^i, \ldots, t_k^i \} \times \prod_{j=0}^{K} (PC_i \times AP) \right)$$

- Transition relation between states: $R \subseteq S \times S$. 
Translation Algorithm

@Perm(\texttt{requires}="\textit{full(this) in A}", \texttt{ensures}="\textit{full(this) in B}")

\begin{align*}
\text{StartMethod(} & s: \text{GlobalState}, t: \text{GlobalTypestate}, r^i_j: \text{Reference}, \\
& m: \text{Method}_i, \left( \left( r^{i_0}_{i_0}, ts^{k_0}_{i_0}, ap_0 \right), \left( r^{i_1}_{i_1}, ts^{k_1}_{i_1}, ap_1 \right) \right) : \text{Triple} \times \text{Triple} ) \\
\text{guard } \leftarrow & \ s[i][j].ap \neq \bot \land s[i][j].pc = (\text{notExe}, \cdot) \land t[i_0] = ts^{i_0}_{i_0} \land \\
& \text{Comp} \left( s[i_0][j_0].ap, ap_0 \right) \land \text{Comp} \left( s[i_1][j_1].ap, ap_1 \right) \\
\text{update } \leftarrow & \ s'[i][j].pc = (\text{Exe}, m) \land \text{ChangePermission} \left( s[i_0][j_0], ap_0 \right) \\
\text{return } & \text{guard } \Rightarrow \text{update}
\end{align*}
Translation Algorithm (Co.)

EndMethod( s: GlobalState, t: GlobalTypestate, \( r^j_i \): Reference, 
\( m \): Method_i, \((r^0_i, ts^0_i, ap_0), (r^1_i, ts^1_i, ap_1)\)): Triple × Triple

\[ \text{guard } \leftarrow s[i][j].pc = (\text{Exe}, m) \]

\[ \text{update } \leftarrow t'[i] = ts^1_{i1} \land s'[i][j].ap = ap_1 \land s'[i][j].pc = (\text{notExe}, m) \land \text{ChangePermission}(s[i][j].ap, ap_1) \]

\[ \text{return guard } \Rightarrow \text{update} \]
Model Structure

- Variables Declarations
- Variables Initialisation
- Transitions Relations
- Create Alias
- CTL Properties
CTL Properties

- Sink States (Deadlock)
  
  \[ \text{deadlock} : \neg \exists X (\text{true}) \]

- Typesates Transition Matrix (Graph)
  
  \[ \text{adjacent}_i(t_1, t_2) : \text{state}_i = t_1 \land \exists X (\text{state}_i = t_2) \]

- Concurrency
  
  \[ \text{concurrent}_i(m_1, m_2) : \]
  \[ \exists X \left( pc_{i}^{j_1} = (m_1, \text{Exe}) \land pc_{i}^{j_2} = (m_2, \text{Exe}) \right) \]

- Methods Reachability
  
  \[ \text{method}^j_i : \exists X \left( pc^j_i = (m_i, \text{Exe}) \right) \]
Demonstration through Eclipse
Conclusion

- Implementation contains
  - Simple Plural Specification
  - Specification with @Cases
  - Specification with Parameters
  - Specification with @Refine Clause
  - Specification with *(and) Clause
- Implementation does not contain
  - Abstraction for state invaraiants
Conclusion (co.)

- New window to evaluate Plural specifications
- Enhances existing strengths of Plural
- Approach is scalable
- Practically useful evaluated through MTTS
- Find numerous error types, state and method reachability