Structure-aware computation of predicate abstraction

A. Cimatti, J. Dubrovin, T. Junntila, M. Roveri
Fondazione Bruno Kessler, Trento, Italy
Helsinki Institute of Technology, Finland
Predicate abstraction

- Concrete program $C$ over states $S$
- Predicates $\Psi_i$ induce partition over $S$
- Each partition is a state of the abstract program
- Transitions in abstract space
  - from $as$ to $as'$ iff $c$-transition from $cs$ to $cs'$, with $cs$ in $as$, and $cs'$ in $as'$
Predicate abstraction: symbolic view

- Concrete state as assignment to $X$ variables
  - booleans, bit vectors, reals, integers, ...
- Concrete program as SMT formula $CR(X, X')$
- Abstract state as assignment to boolean variables $P_i$
- Predicates as SMT formulae $\Psi_i(X)$
- Abstraction function $\text{Abstr}(X, X' \ P P')$ as $\bigwedge_i P_i \leftrightarrow \Psi_i(X)$

Computing predicate abstraction:
- Obtain a boolean representation for $AR(P, P')$
- Amenable to symbolic model checking

$\begin{align*}
AR(P, P') &= \exists X X'. (CR(X, X') \land \bigwedge_i P_i \leftrightarrow \Psi_i(X) \\
& \quad \land \bigwedge_i P'_i \leftrightarrow \Psi_i(X') )
\end{align*}$
From Q-SMT to Boolean

\[ \exists X X' \]

\[ \Phi(X X' P P') \]

\[ \Phi_B(P P') \]

- Predicate Abstraction
  - at the core of many verification approaches
  - often a bottleneck
Avoid Monolithic Computation

\[ \exists X, X' \in \Phi(XX'PP') \]

Reduce

Abstract

Abstract

Abstract

\[ \Phi_B(PP') \]

Structure-aware abstraction

FMCAD'09, Austin, TX
Structure-aware predicate abstraction

• New procedure for predicate abstraction

• Exploits the available problem structure

• At the high level
  – structure of system being abstracted
  – modules, scope of variables, nature of transitions

• At the low level
  – structure of quantified formula
  – reduce scope of quantification
High level framework

- System structured in several components
- Asynchronously composed via interleaving
- Transitions:
  - local transitions
  - synchronizing transitions
  - timed transitions

Invariants: $x$ in $[10, 20]$
SMT: $10 \leq x \& x \leq 20$

Flow condition: $\text{der}(x)$ in $[1.1, 1.3]$
SMT: $x + 1.1 \cdot \delta \leq x' \& x' \leq x + 1.3 \cdot \delta$

Global: the same $\delta$ for all components!
Predicate abstraction procedure

• Ingredients
  – disjunctively partitioning the concrete program
  – inlining
  – clustering
  – blocking and restricting models
  – value sampling
Disjunctive Partitioning

- \( \text{AR}(P P') = \exists X X'.(\text{CR}(X X') \land \text{Abstr}(X X' P P')) \)
- Present CR as disjunction of
  - \( V_m V_l E_t(X X') \) local transitions
  - \( V_{m,m'} E_{\sigma}(X X') \) synchronizing transitions
  - \( V_l E_{\delta}(X X') \) timed transitions
- Distribute \( \text{Abstr}(X X' P P') \) over disjuncts
- Push Quantification inside disjunction
- \( \text{AR}(P P') = \)
  - \( (V_m V_l \exists X X'.(E_t(X X') \land \text{Abstr}(X X' P P')))) \lor \)
  - \( (V_{m,m'} \exists X X'.(E_{\sigma}(X X') \land \text{Abstr}(X X' P P')))) \lor \)
  - \( V_l \exists X X'.(E_{\delta}(X X') \land \text{Abstr}(X X' P P'))) \)
Abstracting one transition

- During transitions, several components may not change
- In local transitions
  - only active process is modified
  - \( \text{loc}' = \text{loc}, \ x' = x, \ldots \)
- Synchronizing transitions
  - similarly, only active processes change
- Timed transitions
  - discrete locations do not change
- Lots of potential for inlining
Rules for inlining

- $\exists X. (\beta \land (u=\alpha))$ rewrites to $\exists X. (\beta[u/\alpha])$
  - where $u$ in $X$, and not in $\alpha$

- $\exists X. (\beta \land (q \leftrightarrow \alpha))$ rewrites to $(q \leftrightarrow \alpha) \land \exists X. (\beta[q/\alpha])$
  - where $\alpha$ propositional, and $q$ not in $\alpha$

- $\exists X. (\beta \land (\gamma \leftrightarrow \alpha))$ rewrites to $\exists X. (\beta[\gamma/\alpha]) \land (\gamma \leftrightarrow \alpha))$
  - where $\alpha$ propositional but $\gamma$ has vars in $X$
Practical Limitations

• Variable in one component may be referred to in flow conditions of other components
  – this indirectly influences its behaviour.

• Predicates can introduce correlations that are not directly present in the original system
  – e.g. \((x + y < 10)\) connects \(x\) and \(y\)
Clustering

- $\exists X. (\Phi_1(X_1 P) \land \Phi_2(X_2 P) \land \ldots \land \Phi_n(X_n P))$
- Each variable in $X$ occurs in at most one of the clusters $X_i$
- Each cluster can be dealt with independently
- Trade one big quantification for many (hopefully smaller) quantifications

$(\exists X_1.\Phi_1(X_1 P)) \land (\exists X_2.\Phi_2(X_2 P)) \land \ldots \land (\exists X_n.\Phi_n(X_n P))$
Blocking and Restricting Models

- When computing $\Phi_B(P) \lor \exists X.\Phi(XP)$
- Replace $\exists X.\Phi(XP)$ with $\exists X.(\neg\Phi_B(P) \land \Phi(XP))$
- Rationale
  - boolean reasoning cheaper than SMT reasoning
  - models in $\Phi_B$ have already been visited
  - force exploration to other models within $\neg\Phi_B$

- When computing
  - $\Phi_{B0}(P) \land \exists X_1.\Phi_1(X_1P) \land \exists X_2.\Phi_2(X_2P) \land \ldots \land \exists X_n.\Phi_n(XnP)$
- We can use previously computed conjuncts to prune quantification
  - $\exists X_1.(\Phi_1(X_1P) \land \neg\Phi_{B0}(P))$
  - $\exists X_2.(\Phi_2(X_2P) \land \neg\Phi_{B01}(P))$
  - $\exists X_3.(\Phi_3(X_3P) \land \neg\Phi_{B012}(P))$
- Restrict to models still worth exploration
Variable Sampling

- "Quasi clustering": a single $w$ prevents clustering
  - $\exists X.(\Phi_1(w \, X_1 \, P) \land \Phi_2(w \, X_2 \, P) \land \ldots \land \Phi_n(w \, X_n \, P))$
- Pick one value $c$ for $w$, replace, and cluster
  - $\exists X\backslash w.(\Phi_{1,w/c}(X_1 \, P) \land \Phi_{2,w/c}(X_2 \, P) \land \ldots \land \Phi_{n,w/c}(X_n \, P)$
- Result: underapproximation $\Phi_{w/c}(P)$
  - computed one cofactor with respect to $w = c$
  - we have to cover the case $w \neq c$
  - $\exists X.(w \neq c \land \Phi_1(w \, X_1 \, P) \land \Phi_2(w \, X_2 \, P) \land \ldots \land \Phi_n(w \, X_n \, P))$
- The process can be iterated
  - need to block already covered models
  - need to find a suitable sequence of instantiations
Sampling-driven quantification

SamplingAllSMT(\(\Phi, X, W\)) {
    res := False;
    (sat, mu) := SMTSolve(\(\Phi\));
    while sat do
        c := PickValue(mu, W);
        new := AllSMT(not res and \(\Phi[W / c]\));
        res := res or new;
        (sat, mu) := SMTSolve(\(\Phi and not res\));
    end while
    return res;
}
Implementation

• Extended NuSMV
  – empowered with SMT functionalities
  – types: reals, integers, bit-vectors, …
• MathSAT SMT solver used as backend
• High level simplifications
  – network of automata
  – python script to generate disjunctive partitioned representation
• Low level simplifications as rewriter over quantified formulae
• Abstraction based on AllSMT version of MathSAT
Experimental Set up

• Two classes of problems
  – from HyTech distribution
  – randomly generated networks of automata

• Compared Algorithms
  – mono
  – + partitioning
  – + clustering
  – + v-sampling
## Results on Hytech models

| Model            | $|\bar{P}|$ | $|\bar{V}|$ | disj | computation time (s) | sampling |
|------------------|---------|---------|-----|----------------------|----------|
|                  |         |         |     | monol.   | partit.  | clust.   | sampl.  |
|                  |         |         |     |          |         |          |         |
| active           | 34      | 5       | 27  | 54.626   | 18.847  | 2.410    | 0.937   |
| active-trace     | 34      | 7       | 27  | 51.781   | 22.171  | 2.473    | 0.952   |
| audio            | 30      | 6       | 15  | 13.826   | 4.547   | 0.448    | 0.442   |
| audio-timing     | 29      | 7       | 15  | 10.910   | 3.915   | 0.947    | 0.690   |
| billiard-timed   | 25      | 3       | 5   | 0.910    | 0.732   | 0.732    | 1.044   |
| dist-controller  | 8       | 7       | 12  | 0.320    | 0.232   | 0.195    | 0.147   |
| grc-ver          | 24      | 5       | 11  | 33.068   | 19.599  | 10.421   | 0.455   |
| new-grc          | 22      | 5       | 11  | 38.649   | 17.840  | 7.395    | 0.383   |
| railroad         | 16      | 3       | 8   | 0.170    | 0.140   | 0.131    | 0.112   |
| reactor-clock    | 19      | 4       | 5   | 0.181    | 0.133   | 0.069    | 0.050   |
| reactor-rect     | 17      | 4       | 5   | 0.132    | 0.112   | 0.051    | 0.045   |
Results on Random LHA's
Related Work

- Imprecise techniques
  - Cartesian Abstraction
- Boolean Quantification
  - BDD-based
  - SAT-based
- Monolithic SMT-based predicate abstraction
  - AllSMT [CAV06]
  - BDD + SMT [FMCAD07]

- Software model checking: BLAST, SATABS
  - Partitioning transition by transition in CFG
  - Forward image computations by inlining unmodified variables

- Avoid abstraction computation
  - Directly compute abstract violations [FM09]
  - No need for AllSMT functionality
Conclusions

• A structure-aware procedure for the exact computation of predicate abstraction

• Exploit high level structure
  – transition partitioning
  – variable scope

• Exploit low level structure
  – formula quantification, clustering
  – value sampling

• Significant speed-ups
Future Work

• Comprehensive comparison with other methods
  – Experiment with BDD-based abstraction
• Measure impact on CEGAR loop
• Application to post-image computation
  – Reachability in abstract space
• Full incrementality