Parameter Synthesis with IC3

Alessandro Cimatti, Alberto Griggio, Sergio Mover, Stefano Tonetta

Fondazione Bruno Kessler, Trento, Italy

April 11, 2015

SynCoP 2015

This work was presented at FMCAD 2013 [CGMT13]
Motivations

- Context: automatic-design of safety-critical systems
- Systems have parameters (e.g. unknown constants):
  - instantiated only in the implementation
  - represent uncertainty (e.g. environment, robustness of the system)

**Problem:** automatically synthesize parameter valuations that satisfy some requirements
Parameter synthesis - several possible instance

- infinite vs. finite domain parameters
- satisfy invariant vs. LTL properties
- universal vs. existential problem
  (for all executions the property hold vs.
   there exists an execution such that the property does not hold)
- all the valuations vs. some
- optimal vs. no requirements
Settings in this talk

- **infinite** vs. finite domain parameters
  - Real-type parameters
- satisfy **invariant** vs. LTL properties
- **universal** vs. existential problem
  - property holds for all the possible traces
- all the **valuations** vs. some
- **optimal** vs. no requirements
  - we do not focus on optimal
  - (the set found could be further refined, e.g. using OptiMathsat)
Settings in this talk

- **infinite** vs. finite domain parameters
  Real-type parameters
- **satisfy invariant** vs. LTL properties
- **universal** vs. existential problem
  property holds for all the possible traces
- **all the valuations** vs. some
- **optimal** vs. **no requirements**
  we do not focus on optimal
  (the set found could be further refined, e.g. using OptiMathsat)

Synthesize the set of all the parameters valuations that guarantees that an invariant property hold
Several problems can be casted to parameter synthesis, e.g.:

- sensor synthesis for diagnosability
  parameters are sensors, property to fulfill “the system is diagnosable”
  E.g. [Bittner et.al., FMCAD14]

- fault tree generation
  parameters are fault variables, properties are top-level events
We extend IC3 [Bradley, VMCAI11] to perform parameter synthesis:

- IC3 is an invariant verification algorithm
- If infinite state transition system
  - Good representation of sw, real-time and hybrid systems
  - It uses Satisfiability Modulo Theories (SMT)
- We exploit some of the IC3 features:
  - It is easy to use in an incremental way
  - It finds a set of traces
  - The IC3 extension to SMT that we use
1 Synthesis with IC3

2 Results

3 Conclusion
Outline

1 Synthesis with IC3

2 Results

3 Conclusion
$S = \langle U, V, I, T \rangle$ is a parametric transition system

- $U$: set of parameters (of real type),
- $V$: set of variables,
- $I(V \cup U)$: initial states,
- $T(V \cup U \cup V')$: transitions (semantic: parameters never change);
Our starting point [CPR08]

\[ \rho(U) := TRUE, k := 0 \]
\[ S' := \langle V \cup U, I, T \land \rho(U) \land \bigwedge_{u \in U} u' = u \rangle \]

\[ \text{Algorithm:} \]

- iteratively finds a set of **unfeasible** parameter valuations
- uses Bounded Model Checking (BMC) to find a counterexample at the k-th step

\[ \beta(U) := \exists V^0, \ldots, V^k. \text{BMC}(S', k, P) \]
\[ \rho(U) := \rho(U) \land \neg \beta(U) \]
\[ k := k + 1 \]

\[ \text{BMC}(S', k, P) \text{ is satisfiable?} \]
\[ k = K_{\text{max}}? \]

YES (\( \neg P \) reachable in k steps)

NO

YES

NO

\[ \rho(U) \text{ is the set of feasible parameter valuations} \]
Our starting point [CPR08]

\[ \rho(U) := \text{TRUE}, k := 0 \]
\[ S' := \langle V \cup U, I, T \land \rho(U) \land \land_{u \in U} u' = u \rangle \]

\[ \begin{align*}
BMC(S', k, P) \text{ is satisfiable?} \\
\beta(U) &=: \exists V^0, \ldots, V^k. BMC(S', k, P) \\
\rho(U) &=: \rho(U) \land \neg \beta(U) \\
k &=: k + 1
\end{align*} \]

\[ k = K_{\text{max}}? \]

Weaknesses:

- the termination by assuming the existence of a maximum bound \( K_{\text{max}} \)
- quantifier elimination to compute \( \beta(U) \) is a bottleneck.
Disclaimer: I am skipping most of the details of IC3
Disclaimer: I am skipping most of the details of IC3.
IC3 builds a sequence of sets of states (frames) $F_0; F_1; \ldots; F_k$:

- $i \leq k$, $F_i \models P$
- $i < k$, $F_i \rightarrow F_{i+1}$
- $i < k$, $F_i \land T \rightarrow F_{i+1}$
IC3 in a nutshell

Disclaimer: I am skipping most of the details of IC3
IC3 builds a sequence of sets of states (frames) \( F_0; F_1; \ldots; F_k: \)

- \( i \leq k, F_i \models P \)
- \( i < k, F_i \rightarrow F_{i+1} \)
- \( i < k, F_i \land T \rightarrow F_{i+1} \)

How does it work?

- it iterates 2 phases:
  1. Blocking phase: proves that the frame \( F_k \) cannot reach \( \neg P \)
  2. Propagation phase: proves that some facts that hold at \( F_i \) also holds at \( F_{i+1} \)
- it stops if either finds a counterexample or an inductive invariant
IC3 in a nutshell

Disclaimer: I am skipping most of the details of IC3

IC3 builds a sequence of sets of states (frames) $F_0; F_1; \ldots; F_k$:

- $i \leq k$, $F_i \models P$
- $i < k$, $F_i \rightarrow F_{i+1}$
- $i < k$, $F_i \land T \rightarrow F_{i+1}$

How does it work?

- it iterates 2 phases:
  1. **Blocking phase**: proves that the frame $F_k$ cannot reach $\neg P$
  2. Propagation phase: proves that some facts that hold at $F_i$ also holds at $F_{i+1}$

- it stops if either finds a counterexample or an inductive invariant

we are interested in the **blocking phase**, which finds counterexamples to $P$
Can we reach \( s_{k+1} \) from \( F_k \)? (i.e. \( F_k \land \neg s_{k+1} \land T \rightarrow s'_{k+1} \)?)

No: \( s_{k+1} \) cannot be reached from \( F_k \).

Yes: \( s_{k+1} \) can be reached from \( F_k \).

We find a set of predecessors
\[
F_0, F_1, F_2, \ldots, F_{k+1}
\]
are sets of states
\[
\pi = s_0; \ldots; s_k
\]
represents a set of traces!
Can we reach $s_{k+1}$ from $F_k$? (i.e. $F_k \land \neg s_{k+1} \land T \rightarrow s'_{k+1}$?)
Can we reach $s_{k+1}$ from $F_k$? (i.e. $F_k \land \neg s_{k+1} \land T \rightarrow s'_{k+1}$?)

- No: $s_{k+1}$ cannot be reached from $F_k$
Can we reach $s_{k+1}$ from $F_k$? (i.e. $F_k \land \neg s_{k+1} \land T \rightarrow s'_{k+1}$?)

- No: $s_{k+1}$ cannot be reached from $F_k$
- Yes: $s_{k+1}$ can be reached from $F_k$. 
Can we reach $s_{k+1}$ from $F_k$? (i.e. $F_k \land \neg s_{k+1} \land T \rightarrow s'_{k+1}$?)

- No: $s_{k+1}$ cannot be reached from $F_k$
- Yes: $s_{k+1}$ can be reached from $F_k$.

We find a set of predecessors $s_k = \exists V. (F_k \land \neg s_{k+1} \land T \land s'_{k+1})$

$s_k$ is computed with an under-approximated qelim
Can we reach $s_{k+1}$ from $F_k$? (i.e. $F_k \land \neg s_{k+1} \land T \rightarrow s'_{k+1}$?)

- No: $s_{k+1}$ cannot be reached from $F_k$
- Yes: $s_{k+1}$ can be reached from $F_k$.

We find a set of predecessors $s_k = \exists V. (F_k \land \neg s_{k+1} \land T \land s'_{k+1})$

$s_k$ is computed with an under-approximated qelim

$s_0, \ldots, s_k$ are sets of states $\rightarrow \pi = s_0; \ldots; s_k$ represents a set of traces!
IC3 for synthesis - algorithm

\[ \rho(U) := TRUE \]

\[ S' := \langle V \cup U, I, T \land \rho(U) \land \bigwedge_{u \in U} u' = u \rangle \]

\[ IC3(S', P) \]

return safe?

YES

\[ \rho(U) \text{ is the set of feasible parameter valuations} \]

NO

\[ \beta(U) := \exists V^0.s_0 \]

\[ \rho(U) := \rho(U) \land \neg \beta(U) \]

Cheap quantifier elimination

- \( s_0 \) represents a set of states
- All the states in \( s_0 \) can reach \( \neg P \) (they are “bad” states)
- cheap quantifier elimination \( \exists V^0.s_0(V^0, U) \)
Optimizations 1/2 - Incrementality

\[ \rho(U) := \text{TRUE} \]

\[ S' := \langle V \cup U, I, T \land \rho(U) \land \bigwedge_{u \in U} u' = u \rangle \]

\[ \text{IC3}(S', P) \text{ is safe?} \]

YES

\[ \rho(U) \] is the set of feasible parameter valuations

NO

\[ \beta(U) := \exists V. s_0 \]

\[ \rho(U) := \rho(U) \land \neg \beta(U) \]

\[ I := I \land \neg \beta(U) \]

\[ T := T \land \neg \beta(U) \]

- No need to restart from scratch → keep all the previously computed frames
- Exploits the incrementality in the SMT solver
\( \pi = s_0; \ldots; s_k \) represents a set of traces, each \( s_i \) is bad!

we can try several heuristics:

- \( \exists V. s_0 \)
- \( j \leq k, \exists V^0, \ldots, V^j. I \land \bigwedge_{i=0}^{i<j}(T^i \land s_i) \land s_j \)
- \( \exists V^0, \ldots, V^k. I \land \bigwedge_{j=0}^{j<k}(T^j \land \neg P^j) \)

we can trade results’ generality with the cost of the quantifier elimination

in practice: start with \( V. s_0 \) and switch to

\( \exists V^0, \ldots, V^k. I \land \bigwedge_{j=0}^{j<k}(T^j \land \neg P^j) \) after enumerating too many counterexamples
Outline

1. Synthesis with IC3
2. Results
3. Conclusion
Experimental evaluation

- We use linear hybrid automata benchmarks
- Our implementation:
  - uses the \texttt{MathSat} SMT solver
  - done inside the \texttt{nuXmv} model checker
  - it is integrated in \texttt{HyCOMP}, a model checker for hybrid systems

  more on \texttt{HyCOMP} at TACAS

- Competitors:
  - \texttt{Iter-Block-Path (IC3)}: non-incremental version of [CPR08], using IC3 as a “black box”
  - \texttt{RED [Wan05]}: tool for parameter synthesis for linear hybrid automata:
    - it computes all the reachable states of the system
    - symbolic representation of the state space
Results

ParmaIC3

Iter-Block-Path(IC3)

RED

Parameter Synthesis with IC3

Sergio Mover (FBK)

April 11, 2015
Outline

1. Synthesis with IC3
2. Results
3. Conclusion
Conclusion

In this talk we shown a simple extension of IC3 to synthesize parameters

- the approach works for infinite-state transition systems
- it exploits IC3 for incrementality and to have a cheap quantifier elimination
- good performance results

Future works

- directly solve the “universal” problem
- a lot of future works are motivated by recent extensions of IC3:
  - to use predicate abstraction: more efficient algorithm
  - to prove LTL formulas: synthesize parameters that preserve liveness
- find the optimal set of parameter regions (e.g. consider pareto optimality)
- extend the approach to hybrid systems with more complex dynamics
Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta.
Parameter synthesis with ic3.

Alessandro Cimatti, Luigi Palopoli, and Yusi Ramadian.
Symbolic computation of schedulability regions using parametric timed automata.

Farn Wang.
Symbolic parametric safety analysis of linear hybrid systems with bdd-like data-structures.
Parameter synthesis - formal definition

$S = \langle U, V, I, T \rangle$ is a parametric transition system
  - $U$: set of parameters, $V$: set of variables,
  - $I(V \cup U)$: initial states, $T(V \cup U \cup V')$: transitions;

$S_\gamma = \langle V, \gamma(I), \gamma(T) \rangle$: TS induced by the valuation $\gamma$
  - $\gamma$: assigns a value to each parameter in $U$
  - $\gamma(\phi)$: substitute in $\phi$ each occurrence of a parameter with its value
    (e.g. $\gamma = \{ p = 3 \}$, $\gamma(x + p \leq 0) := x + 3 \leq 0$)

Parameter synthesis problem:
  - $P(U, X)$ is an invariant property
  - $\gamma$ is feasible for the property iff $S_\gamma \models \gamma(P)$.
  - find the set $\rho(U)$ of all the feasible parameter valuations.
Tradeoff generality/costs of quantifier elimination

![Graph showing the tradeoff between generality and costs of quantifier elimination. The x-axis represents ParamIC3-basic, and the y-axis represents ParamIC3. The data points are plotted on a logarithmic scale, with red lines indicating equal values.]