AbsSynthe: abstract synthesis from succinct safety specifications

Romain Brenguier, Guillermo A. Pérez, Jean-François Raskin, Ocan Sankur

Université Libre de Bruxelles – Brussels, Belgium
SYNT’14 @ Vienna

July, 2014
Outline

1. Succinct safety specs = Safety games

2. The classic algorithm
   - Main idea
   - The uncontrollable predecessors’ operator

3. A CEGAR algorithm
   - Contributions
   - Abstract game
   - Abstract operators
   - The algorithm

4. Benchmarks & conclusions
What is a succinct safety spec?

In essence: a **boolean network** for a single-output sequential circuit:

- A set of boolean inputs \( X \),
- a set of boolean latches \( L \) with a distinguished error latch \( BAD \in L \).

The circuit defines a boolean function \( f_i \) over \( L \) and \( X \) per latch.
For synthesis, $X$ partitioned into uncontrollable $X_u$ and controllable inputs $X_c$. $X_u$ are chosen by the environment.
Synthesis from a succinct safety spec

For synthesis, $X$ partitioned into uncontrollable $X_u$ and controllable inputs $X_c$. $X_u$ are chosen by the environment.
Synthesis from a succinct safety spec

For synthesis, $X$ partitioned into uncontrollable $X_u$ and controllable inputs $X_c$. $X_u$ are chosen by the environment.

Realizability: does there exist such a controller for $X_c$?
The safety game

- $Q$ is the set of valuations of $L$, $U \subseteq Q$ are the error states
- $\Sigma_u, \Sigma_c$ are valuations of $X_u, X_c$ resp.
- $\delta : Q \times \Sigma_u \times \Sigma_c \rightarrow Q$ defined by circuit $C$
- Game: environment chooses $\sigma$ and controller responds with $\tau$

Example: $L = \{l_0, BAD\}$, $\Sigma_u = \{r, \bar{r}\}$, $\Sigma_c = \{g, \bar{g}\}$. 
The safety game

- $Q$ is the set of valuations of $L$, $\mathcal{U} \subseteq Q$ are the error states
- $\Sigma_u, \Sigma_c$ are valuations of $X_u, X_c$ resp.
- $\delta : Q \times \Sigma_u \times \Sigma_c \rightarrow Q$ defined by circuit $C$
- Game: environment chooses $\sigma$ and controller responds with $\tau$

Example: $L = \{l_0, BAD\}$, $\Sigma_u = \{r, \bar{r}\}$, $\Sigma_c = \{g, \bar{g}\}$.
The safety game

- $Q$ is the set of valuations of $L$, $\mathcal{U} \subseteq Q$ are the error states
- $\Sigma_u, \Sigma_c$ are valuations of $X_u, X_c$ resp.
- $\delta : Q \times \Sigma_u \times \Sigma_c \rightarrow Q$ defined by circuit $\mathcal{C}$
- Game: environment chooses $\sigma$ and controller responds with $\tau$

Example: $L = \{l_0, BAD\}$, $\Sigma_u = \{r, \bar{r}\}$, $\Sigma_c = \{g, \bar{g}\}$. 
1. Succinct safety specs = Safety games

2. The classic algorithm
   - Main idea
   - The uncontrollable predecessors’ operator

3. A CEGAR algorithm
   - Contributions
   - Abstract game
   - Abstract operators
   - The algorithm

4. Benchmarks & conclusions
Based on the **Reachability Game** played on the graph \( \langle Q, \Sigma_u, \Sigma_c, \delta, U \rangle \):

1. Define an **uncontrollable predecessors** operator \( \text{UPRE} \).
2. Compute the least fixpoint of \( \text{UPRE} \) starting from the error states (call this \( W_u \)).
3. From \( W_c = Q \setminus W_u \) **controller** can respond to a given \( \sigma \) with any \( \tau \) which ensures she stays in \( W_c \).
UPRE: definition by example

UPRE(S) is the set of states from which environment can force to reach S.
If $\Sigma_u = \{\sigma_0, \overline{\sigma_0}\}$ and $\Sigma_c = \{\tau_0, \overline{\tau_0}\}$, then UPRE(d, e) = ???

![Diagram](image.png)
UPRE(S) is the set of states from which \textbf{environment} can force to reach S.

If $\Sigma_u = \{\sigma_0, \overline{\sigma_0}\}$ and $\Sigma_c = \{\tau_0, \overline{\tau_0}\}$, then $\text{UPRE}(d, e) = \{c\}$.
The classic algorithm

Based on the Reachability Game played on the graph $\langle Q, \Sigma_u, \Sigma_c, \delta, \mathcal{U} \rangle$:

1. Define an uncontrollable predecessors operator UPRE.
2. Compute the least fixpoint of UPRE starting from the error states (call this $W_u$).
3. From $W_c = Q \setminus W_u$ controller can respond to a given $\sigma$ with any $\tau$ which ensures she stays in $W_c$. 
The classic algorithm

Based on the Reachability Game played on the graph $\langle Q, \Sigma_u, \Sigma_c, \delta, U \rangle$:

1. Define an uncontrollable predecessors operator UPRE.
2. Compute the least fixpoint of UPRE starting from the error states (call this $W_u$).
3. From $W_c = Q \setminus W_u$ controller can respond to a given $\sigma$ with any $\tau$ which ensures she stays in $W_c$. 
How do we compute UPRE?

Using BDDs...

1. Either compute a transition relation

\[ T(L, X_u, X_c, L') = \bigwedge_{l \in L} l' \iff f_l(X_u, X_c, L) \]

and then set \( \text{UPRE}(S) = \exists X_u, \forall X_c, \exists L' : T(L, X_u, X_c, L') \land S(L') \); or

2. for deterministic systems we can avoid computing \( T \) and just substitute \( f_l \) for each \( l \) in \( S \)

\[ \text{UPRE}(S) = \exists X_u, \forall X_c : S(L')[l' \leftarrow f_l(X_u, X_c, L)]_{l \in L}. \]

[Coudert et al., 1990, Coudert et al., 1991]
How do we compute UPRE?

Using BDDs...

1. Either compute a transition relation:

\[ T(L, X_u, X_c, L') = \bigwedge_{l \in L} l' \iff f_l(X_u, X_c, L) \]

and then set \( \text{UPRE}(S) = \exists X_u, \forall X_c, \exists L' : T(L, X_u, X_c, L') \land S(L') \); or

2. for deterministic systems we can avoid computing \( T \) and just substitute \( f_l \) for each \( l \) in \( S \)

\[ \text{UPRE}(S) = \exists X_u, \forall X_c : S(L')[l' \leftarrow f_l(X_u, X_c, L)]_{l \in L}. \]

Computing \( T \) is sometimes too costly (time and size).

---

\(^1\text{Coudert et al., 1990, Coudert et al., 1991}\)
Outline

1. Succinct safety specs = Safety games

2. The classic algorithm
   - Main idea
   - The uncontrollable predecessors’ operator

3. A CEGAR algorithm
   - Contributions
   - Abstract game
   - Abstract operators
   - The algorithm

4. Benchmarks & conclusions
Contributions

We improve on a CEGAR-based approach [de Alfaro and Roy, 2010].

1. Use information from the computation of the over-approx of UPRE to
   - over-approx reachable states fixing winning strategies of environment
   - restrict the uncontrollable actions we need to check to compute UPRE.

2. Use substitution (BDD composition) to avoid computing an abstract transition relation (though post over-approx’d).

3. Simple heuristic for choosing new predicates to refine the abstract game without concrete UPRE.
Contributions

We improve on a CEGAR-based approach [de Alfaro and Roy, 2010].

1. Use information from the computation of the over-approx of UPRE to
   - over-approx reachable states fixing winning strategies of environment
   - restrict the uncontrollable actions we need to check to compute UPRE.

2. Use substitution (BDD composition) to avoid computing an abstract transition relation (though post over-approx’d).

3. Simple heuristic for choosing new predicates to refine the abstract game without concrete UPRE.
Contributions

We improve on a CEGAR-based approach [de Alfaro and Roy, 2010].

1. Use information from the computation of the over-approx of UPRE to
   - over-approx reachable states fixing winning strategies of environment
   - restrict the uncontrollable actions we need to check to compute UPRE.

2. Use substitution (BDD composition) to avoid computing an abstract transition relation (though post over-approx’d).

3. Simple heuristic for choosing new predicates to refine the abstract game without concrete UPRE.
Contributions

We improve on a CEGAR-based approach [de Alfaro and Roy, 2010].

1. Use information from the computation of the over-approx of UPRE to
   - over-approx reachable states fixing winning strategies of environment
   - restrict the uncontrollable actions we need to check to compute UPRE.

2. Use substitution (BDD composition) to avoid computing an abstract transition relation (though post over-approx’d).

3. Simple heuristic for choosing new predicates to refine the abstract game without concrete UPRE.
Outline

1. Succinct safety specs = Safety games

2. The classic algorithm
   - Main idea
   - The uncontrollable predecessors’ operator

3. A CEGAR algorithm
   - Contributions
   - Abstract game
     - Abstract operators
     - The algorithm

4. Benchmarks & conclusions
Q is exponential w.r.t. \( L \), so let us “simplify” the game...

Example: \( L = \{ l_0, l_1, l_{BAD} \} \).
Example of an abstract game

Q is exponential w.r.t. L, so let us “simplify” the game...

- \( Q^a \) defined by predicates \( p_U = l_{BAD}, p_I = \neg(l_0 \lor l_1 \lor l_{BAD}), p_0 = l_0 \)

Example: \( L = \{l_0, l_1, l_{BAD}\} \).
Example of an abstract game

$Q$ is exponential w.r.t. $L$, so let us “simplify” the game...

- $Q^a$ defined by predicates $p_U = l_{BAD}$, $p_I = \neg (l_0 \lor l_1 \lor l_{BAD})$, $p_0 = l_0$
- $\Delta^a$ over-approximates $\delta$

Example: $L = \{l_0, l_1, l_{BAD}\}$. 
Abstract game

Some remarks:

- We require the initial state be distinguishable and $U^a$ to contain $U$.
- The partition of $Q$ is done (mainly) via localization reduction (only $p_R$, $p_I$, $p_U$ are real predicates).
Abstract UPRE operators

$P$ is set of predicates defining $Q^a$. $T^a$ is computed as expected from $T$.

**Definition (Two UPRE operators)**

Given $S^a \subseteq Q^a$ let

- $\overline{\text{UPRE}}_a(S^a) = \exists X_u, \forall X_c, \exists P' : T^a(P, X_u, X_c, P') \land S^a(P')$,
- $\underline{\text{UPRE}}_a(S^a) = \exists X_u, \forall X_c, \forall P' : T^a(P, X_u, X_c, P') \Rightarrow S^a(P')$.

In fact, one can again avoid computing $T^a$ using substitution.

**Lemma (Over- and under-approximating UPRE)**

$\gamma(\overline{\text{UPRE}}_a(U^a)) \subseteq \text{UPRE}^*(U) \subseteq \gamma(\underline{\text{UPRE}}_a(U^a))$. 
If $\Sigma_u = \{\sigma_0, \overline{\sigma_0}\}$ and $\Sigma_c = \{\tau_0, \overline{\tau_0}\}$, then

- $\text{UPRE}_a(\{\{d, e\}\}) = \text{??}$
- $\text{UPRE}_a(\{\{d, e\}\}) = \text{??}$
Abstract UPRE: definition by example

If $\Sigma_u = \{\sigma_0, \overline{\sigma_0}\}$ and $\Sigma_c = \{\tau_0, \overline{\tau_0}\}$, then

- $\text{UPRE}_a(\{\{d, e\}\}) = \{\{b, c\}\}$
- $\text{UPRE}_a(\{\{d, e\}\}) = \{\{b, c\}\}$

\[\text{DISJ of all edges} = \sigma_0 \lor \overline{\tau_0}\]
Abstract UPRE: definition by example

If $\Sigma_u = \{\sigma_0, \overline{\sigma_0}\}$ and $\Sigma_c = \{\tau_0, \overline{\tau_0}\}$, then

- $\overline{\text{UPRE}}_a(\{\{d, e\}\}) = \{\{b, c\}\}$
- $\overline{\text{UPRE}}_a(\{\{d, e\}\}) = \{}$

\[\text{DISJ of all edges} = \overline{\sigma_0} \lor \tau_0\]

\[\text{DISJ of all edges} = \sigma_0 \land \tau_0\]
Based on the abstract game $\langle Q^a, q_i^a, \Sigma_u, \Sigma_c, \Delta^a, \mathcal{U}^a \rangle$ and an over-approximation of the reachable states $R^a$:

1. If $q_i^a \in \overline{\text{UPRE}}^*_a(\mathcal{U}^a)$ environment wins,
2. if $q_i^a \not\in \overline{\text{UPRE}}^*_a(\mathcal{U}^a)$ controller wins,
3. else we do not know who wins $G$... add a new “useful” single-latch predicate to $P$ and repeat.

**Does it terminate?**

Eventually all latches are added, so we converge to the original game.
Assume $q_i^a \notin \text{UPRE}_a^*(\mathcal{U}^a)$ and $q_i^a \in \text{UPRE}_a^*(\mathcal{U}^a)$.

1. Extract a winning non-deterministic strategy of environment $\Lambda^a : Q^a \to \mathcal{P}(\Sigma_u)$,

2. this defines a non-det strategy for him in the original game $\Lambda : Q \to \mathcal{P}(\Sigma_u)$.

**Theorem (All of his winning strats)**

If $\lambda$ is a winning strategy for environment in $G$, then $\lambda$ is “included” in $\Lambda$. 
Contribution: strategies winning for environment

This allows for two nice optimizations!

Corollary (Over-approx reachable and restrict UPRE)

1. If $q^a_i \in \overline{\text{UPRE}}^*_a(U^a)$ then we can restrict our search to states reachable if environment plays $\land^a(P, X_u)$,

2. and we can replace UPRE by

$$\text{UPRE}_\land(S) = \exists X_u, \forall X_c, \exists L' : T(L, X_u, X_c, L') \land \land(L, X_u) \land S(L')$$

which takes less uncontrollable inputs into account.
Look for a strategy of environment in the abstract game.
1. Look for a strategy of environment in the abstract game.
Look for a strategy of environment in the abstract game.
Forward exploration

1. Look for a strategy of environment in the abstract game.
2. Ignore all states not reachable in original game via these strategies.

\[ \mu X. (U^a \cup \overline{UPRE}_a(X)) \cap R^a \]
Contribution: strategies winning for environment

This allows for two nice optimizations!

Corollary (Over-approx reachable and restrict UPRE)

1. If \( q^a_i \in \overline{\text{UPRE}}^*_a(U^a) \) then we can restrict our search to states reachable if environment plays \( \Lambda^a(P, X_u) \).

2. And we can replace UPRE by

\[
\text{UPRE}_\Lambda(S) = \exists X_u, \forall X_c, \exists L' : T(L, X_u, X_c, L') \land \Lambda(L, X_u) \land S(L')
\]

which takes less uncontrollable inputs into account.
Contribution: when is a latch “useful”?

We don’t have a unique answer :-(

Definition (Interesting and useful latches)

Given $U^a$ and current visible latches,

1. we consider a latch $l$ interesting if $l \not\Rightarrow U^a$ and $\neg l \not\Rightarrow U^a$; and
2. we say an interesting latch is useful if there is some already visible latch $v$ such that $f_v(L, X_u, X_c)$ depends on $l$.

The idea is...

The newly visible latch will hopefully make $\Delta^a$ more closely resemble the original $\delta$. 
Contribution: when is a latch “useful”? 

We don’t have a unique answer :-(

Definition (Interesting and useful latches)

Given $U^a$ and current visible latches, 

1. we consider a latch $l$ interesting if $l \not\Rightarrow U^a$ and $\neg l \not\Rightarrow U^a$; and

2. we say an interesting latch is useful if there is some already visible latch $v$ such that $f_v(L, X_u, X_c)$ depends on $l$.

The idea is... 

The newly visible latch will hopefully make $\Delta^a$ more closely resemble the original $\delta$. 

Brenguier, Pérez, Raskin, Sankur (ULB) 
AbsSynthe 
July, 2014
abs_synth\( (G, G^a, R^a) \)

\[
\begin{align*}
1 & \quad w_u := \mu X. (U^a \cup \text{UPRE}_a(X)) \cap R^a; \\
2 & \quad \text{if } q_i^a \in w_u \text{ then return not controllable}; \\
3 & \quad \text{prev} := \emptyset; \\
4 & \quad \text{while } R^a \neq \text{prev} \text{ do} \\
5 & \quad \quad \text{prev} := R^a; \\
6 & \quad \quad W_u := \mu X. (w_u \cup \text{UPRE}_a(X)) \cap R^a; \\
7 & \quad \quad \text{if } q_i^a \not\in W_u \text{ then return controllable}; \\
8 & \quad \quad \Lambda^{env} := \text{non-det strategy defined by } (w_u); \\
9 & \quad \quad R^a := \mu X. (q_i^a \cup \text{post}(X, \Lambda^{env})) \cap R^a; \\
10 & \quad \text{end} \\
11 & \quad w'_u := (\text{UPRE}_{\gamma(\Lambda^{env})}(\gamma(w_u))) \cap \gamma(R^a); \\
12 & \quad \text{if } w'_u \subseteq \gamma(w_u) \text{ then return controllable}; \\
13 & \quad Q^a_2 := \text{refine}(Q^a, w'_u \cup \gamma(w_u), \gamma(R^a)); \\
14 & \quad U^a_2 := \alpha_2(w'_u \cup \gamma(w_u)); \\
15 & \quad \text{return abs_synth}(G, G^a_2, \alpha_2(\gamma(R^a))); \\
\end{align*}
\]
abs_synth\((G, G^a, R^a)\)

1. \(w_u := \mu X. (\mathcal{U}^a \cup \text{UPRE}_a(X)) \cap R^a;\)
2. \(\text{if } q_i^a \in w_u \text{ then return not controllable;}\)
3. \(\text{prev} := \emptyset;\)
4. \(\text{while } R^a \neq \text{prev} \text{ do}\)
   5. \(\text{prev} := R^a;\)
   6. \(W_u := \mu X. (w_u \cup \text{UPRE}_a(X)) \cap R^a;\)
   7. \(\text{if } q_i^a \not\in W_u \text{ then return controllable;}\)
   8. \(\Lambda^{env} := \text{non-det strategy defined by } (w_u);\)
   9. \(R^a := \mu X. (q_i^a \cup \text{post}(X, \Lambda^{env})) \cap R^a;\)
10. \(\text{end}\)
11. \(w'_u := (\text{UPRE}_{\gamma(\Lambda^{env})(\gamma(w_u))) \cap \gamma(R^a));\)
12. \(\text{if } w'_u \subseteq \gamma(w_u) \text{ then return controllable;}\)
13. \(Q^a_2 := \text{refine}(Q^a, w'_u \cup \gamma(w_u), \gamma(R^a));\)
14. \(\mathcal{U}^a_2 := \alpha_2(w'_u \cup \gamma(w_u));\)
15. \(\text{return abs_synth}(G, G^a_2, \alpha_2(\gamma(R^a)));\)
\texttt{abs\_synth}(G, G^a, R^a)

1. \( w_u := \mu X. (U^a \cup \text{UPRE}_a(X)) \cap R^a. \)
2. \textbf{if} \( q_i^a \in w_u \) \textbf{then} \textbf{return} not controllable;
3. \( \text{prev} := \emptyset; \)
4. \textbf{while} \( R^a \neq \text{prev} \) \textbf{do}
   5. \( \text{prev} := R^a; \)
   6. \( W_u := \mu X. (w_u \cup \text{UPRE}_a(X)) \cap R^a; \)
   7. \textbf{if} \( q_i^a \notin W_u \) \textbf{then} \textbf{return} controllable;
   8. \( \Lambda^{env} := \text{non-det strategy defined by } (w_u); \)
   9. \( R^a := \mu X. (q_i^a \cup \text{post}(X, \Lambda^{env})) \cap R^a; \)
5. \textbf{end}
11. \( w'_u := (\text{UPRE}_{\gamma}(\Lambda^{env})(\gamma(w_u))) \cap \gamma(R^a)); \)
12. \textbf{if} \( w'_u \subseteq \gamma(w_u) \) \textbf{then} \textbf{return} controllable;
13. \( Q^2_a := \text{refine}(Q^a, w'_u \cup \gamma(w_u), \gamma(R^a)); \)
14. \( U^2_a := \alpha_2(w'_u \cup \gamma(w_u)); \)
15. \textbf{return} \texttt{abs\_synth}(G, G^a_2, \alpha_2(\gamma(R^a)));
abs_synth(\(G, G^a, R^a\))

1. \(w_u := \mu X. (\mathcal{U}^a \cup \text{UPRE}_a(X)) \cap R^a;\)
2. \(\text{if } q^a_i \in w_u \text{ then return not controllable;}\)
3. \(\text{prev := } \emptyset;\)
4. \(\text{while } R^a \neq \text{prev do}\)
   5. \(\text{prev := } R^a;\)
   6. \(W_u := \mu X. (w_u \cup \text{UPRE}_a(X)) \cap R^a;\)
   7. \(\text{if } q^a_i \notin W_u \text{ then return controllable;}\)
   8. \(\Lambda^\text{env} := \text{non-det strategy defined by } (w_u);\)
   9. \(R^a := \mu X. (q^a_i \cup \text{post}(X, \Lambda^\text{env});\)
10. \(\text{end}\)
11. \(w'_u := (\text{UPRE}_{\gamma(\Lambda^\text{env})}(\gamma(w_u))) \cap \gamma(R^a);\)
12. \(\text{if } w'_u \subseteq \gamma(w_u) \text{ then return controllable;}\)
13. \(Q^a_2 := \text{refine} \left( Q^a, w'_u \cup \gamma(w_u), \gamma(R^a) \right);\)
14. \(\mathcal{U}^a_2 := \alpha_2(w'_u \cup \gamma(w_u));\)
15. \(\text{return abs_synth}(G, G^a_2, \bar{\alpha}_2(\gamma(R^a)));\)
abs_synth\((G, G^a, R^a)\)

1. \(w_u := \mu X. (\mathcal{U}^a \cup \text{UPRE}_a(X)) \cap R^a;\)
2. if \(q^a_i \in w_u\) then return not controllable;
3. prev := \(\emptyset;\)
4. while \(R^a \neq \text{prev}\) do
   5.   prev := \(R^a;\)
   6.   \(W_u := \mu X. (w_u \cup \text{UPRE}_a(X)) \cap R^a;\)
   7.   if \(q^a_i \notin W_u\) then return controllable;
   8.   \(\Lambda^{\text{env}} := \text{non-det strategy defined by } (w_u);\)
   9.   \(R^a := \mu X. (q^a_i \cup \text{post}(X, \Lambda^{\text{env}})) \cap R^a;\)
10. end
11. \(w'_u := (\text{UPRE}_{\gamma(\Lambda^{\text{env}})}(\gamma(w_u))) \cap R^a;\)
12. if \(w'_u \subseteq \gamma(w_u)\) then return controllable;
13. \(Q^a_2 := \text{refine}(Q^a, w'_u \cup \gamma(w_u), \gamma(R^a));\)
14. \(\mathcal{U}^a_2 := \alpha_2(w'_u \cup \gamma(w_u));\)
15. return abs_synth\((G, G^a_2, \overline{\alpha}_2(\gamma(R^a)))\);
Some results

(C) FP computation with a precomputed transition relation$^2$;

(C-TL) no transition relation;

(A) CEGAR algo with a precomputed abstract transition relation;

(A-TL) no transition relation (post overapproximated).

$^2$Base implementation from [Bloem et al., 2014]

Figure: Time (in seconds) to check realizability.

Figure: Time (in seconds) for cnt benchmarks.
Thank you for your attention!

If you want to drink download our tool:

https://github.com/gaperez64/AbsSynthe
