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

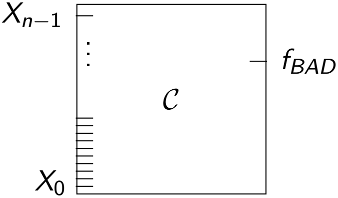
Outline

UNIVERSITÉ LIBRE DE BRUXELLES

What is a succinct safety spec?

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

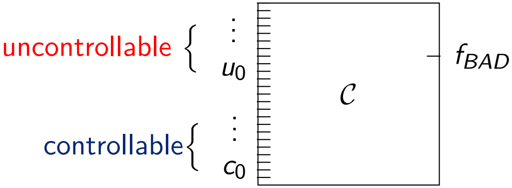
- A set of boolean inputs X,
- a set of boolean latches L with a distinguished error latch $BAD \in L$.



The circuit defines a boolean function f_l over L and X per latch.

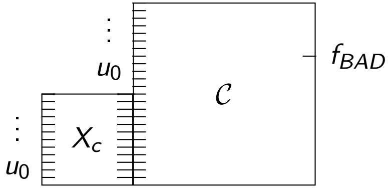
Synthesis from a succinct safety spec

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



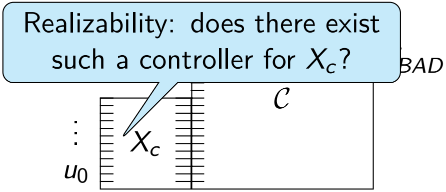
Synthesis from a succinct safety spec

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



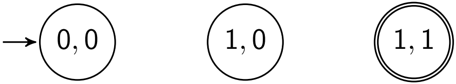
Synthesis from a succinct safety spec

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



The safety game

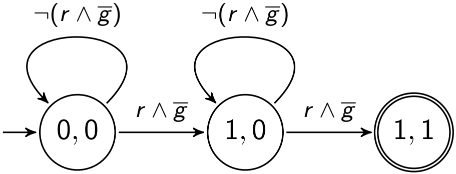
- Q is the set of valuations of L, $\mathcal{U} \subseteq Q$ are the error states
- Σ_u, Σ_c are valuations of X_u, X_c resp.
- $\delta : Q \times \Sigma_u \times \Sigma_c \to Q$ defined by circuit $\mathcal{C}$
- Game: environment chooses σ and controller responds with τ



Example: $L = \{l_0, BAD\}$, $\Sigma_u = \{r, \overline{r}\}$, $\Sigma_c = \{g, \overline{g}\}$.

The safety game

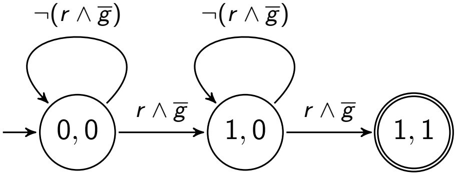
- Q is the set of valuations of L, $\mathcal{U} \subseteq Q$ are the error states
- Σ_u, Σ_c are valuations of X_u, X_c resp.
- $\delta : Q \times \Sigma_u \times \Sigma_c \to Q$ defined by circuit $\mathcal{C}$
- Game: environment chooses σ and controller responds with τ



Example: $L = \{l_0, BAD\}$, $\Sigma_u = \{r, \overline{r}\}$, $\Sigma_c = \{g, \overline{g}\}$.

The safety game

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



Example: $L = \{l_0, BAD\}$, $\Sigma_u = \{r, \overline{r}\}$, $\Sigma_c = \{g, \overline{g}\}$.

Outline

UNIVERSITÉ LIBRE DE BRUXELLES

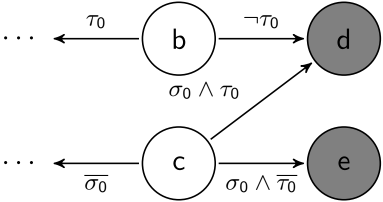
The classic algorithm

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 σ with any τ which ensures she stays in W_c.

UPRE: definition by example

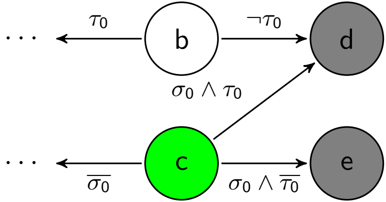
UPRE(S) is the set of states from which environment can force to reach S
If $\Sigma_u = \{\sigma_0, \overline{\sigma_0}\}$ and $\Sigma_c = \{\tau_0, \overline{\tau_0}\}$, then UPRE$(d, e)$ =???



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($d, e$) $=$??? $\{c\}$



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 σ with any τ which ensures she stays in W_c.

UNIVERSITÉ LIBRE DE BRUXELLES

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 σ with any τ which ensures she stays in W_c.

How do we compute UPRE?

Using BDDs...

1. Either compute a transition relation

$$T(L, X_u, X_c, L') = \bigwedge_{l \in L} l' \Leftrightarrow f_l(X_u, X_c, L)$$

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

2. for deterministic systems we can avoid computing T and just substitute f_l for each l in S[1]

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

UNIVERSITÉ LIBRE DE BRUXELLES

[1] [Coudert et al., 1990, Coudert et al., 1991]

How do we compute UPRE?

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

Using BDDs...

1. Either compute a transition relation

$$T(L, X_u, X_c, L') = \bigwedge_{l \in L} l' \Leftrightarrow f_l(X_u, X_c, L)$$

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

2. for deterministic systems we can avoid computing T and just substitute f_l for each l in S[1]

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

[1] [Coudert et al., 1990, Coudert et al., 1991]

Outline

ULB UNIVERSITÉ LIBRE DE BRUXELLES

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].

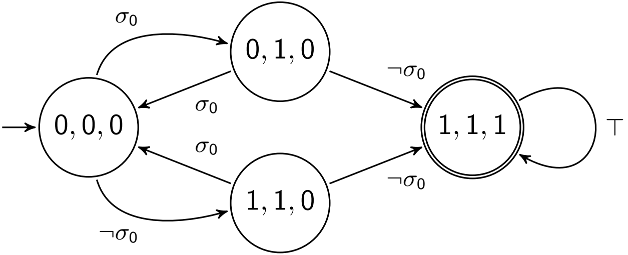
1. Use information from the computation of the over-approx of UPRE to
   - over-approx reachable states fixing winning strategies of environment
   - restrict the uncontrollable actions we need to check to compute UPRE.
2. Use substitution (BDD composition) to avoid computing an abstract transition relation (though post over-approx'd).
3. Simple heuristic for choosing new predicates to refine the abstract game without concrete UPRE.

Outline

UNIVERSITÉ LIBRE DE BRUXELLES

Example of an abstract game

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

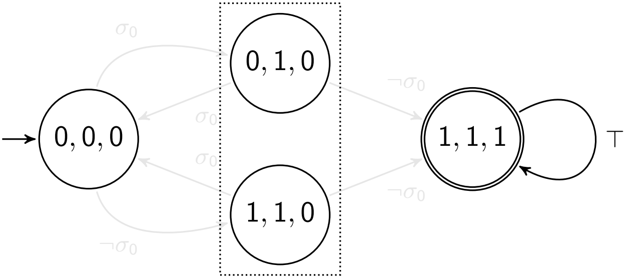


Example: $L = \{l_0, l_1, l_{BAD}\}$.

Example of an abstract game

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

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

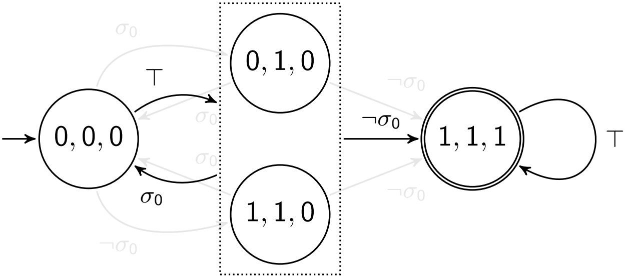


Example: $L = \{l_0, l_1, l_{BAD}\}$.

Example of an abstract game

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

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



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

UNIVERSITÉ LIBRE DE BRUXELLES

Abstract game

Some remarks:

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

UNIVERSITÉ LIBRE DE BRUXELLES

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

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

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

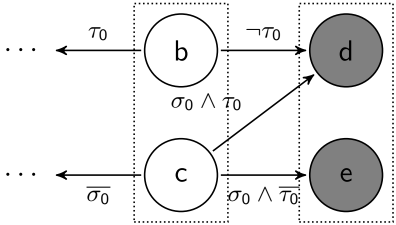
Lemma (Over- and under-approximating UPRE)

$\gamma(\underline{UPRE}^*_a(\mathcal{U}^a)) \subseteq UPRE^*(\mathcal{U}) \subseteq \gamma(\overline{UPRE}^*_a(\mathcal{U}^a))$.

Abstract UPRE: definition by example

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

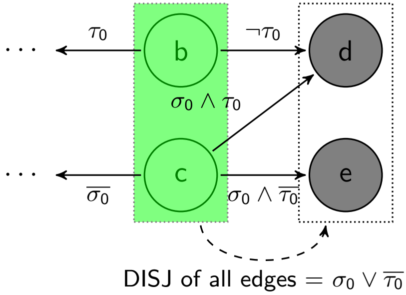
- $\overline{\mathrm{UPRE}}_a(\{\{d, e\}\}) = ???$
- $\underline{\mathrm{UPRE}}_a(\{\{d, e\}\}) = ???$



Abstract UPRE: definition by example

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

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

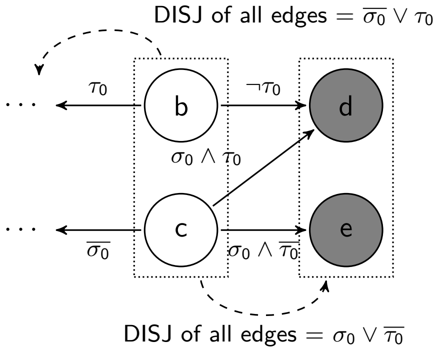


UNIVERSITÉ LIBRE DE BRUXELLES

Abstract UPRE: definition by example

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

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



UNIVERSITÉ LIBRE DE BRUXELLES

A CEGAR algorithm [de Alfaro and Roy, 2010]

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 $q_I^a \in \underline{\mathrm{UPRE}}_a^*(\mathcal{U}^a)$ environment wins,
2. if $q_I^a \notin \overline{\mathrm{UPRE}}_a^*(\mathcal{U}^a)$ 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.

Contribution: strategies winning for environment

Assume $q_I^a \notin \underline{\mathrm{UPRE}}_a^*(\mathcal{U}^a)$ and $q_I^a \in \overline{\mathrm{UPRE}}_a^*(\mathcal{U}^a)$...

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 λ is a winning strategy for environment in G, then λ is "included" in Λ.

UNIVERSITÉ LIBRE DE BRUXELLES

Contribution: strategies winning for environment

This allows for two nice optimizations!

Corollary (Over-approx reachable and restrict UPRE)

1. *If $q_I^a \in \overline{UPRE}_a^*(\mathcal{U}^a)$ then we can restrict our search to states reachable if environment plays $\Lambda^a(P, X_u)$,*
2. *and we can replace UPRE by*

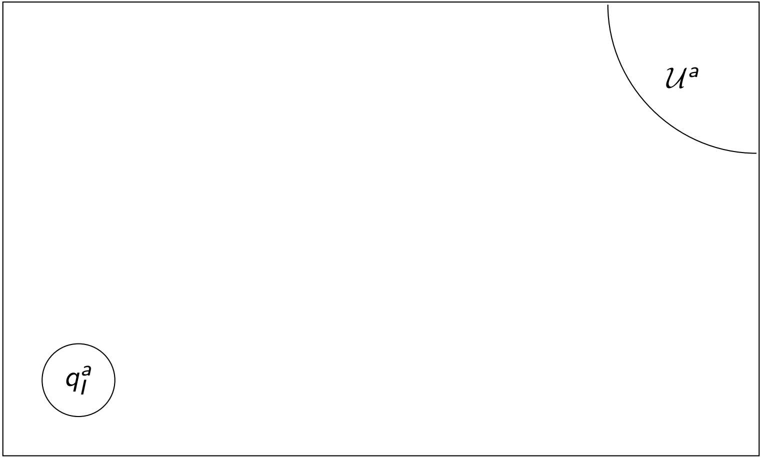
$$UPRE_\Lambda(S) = \exists X_u, \forall X_c, \exists L' : T(L, X_u, X_c, L') \wedge \Lambda(L, X_u) \wedge S(L')$$

which takes less uncontrollable inputs into account.

UNIVERSITÉ LIBRE DE BRUXELLES

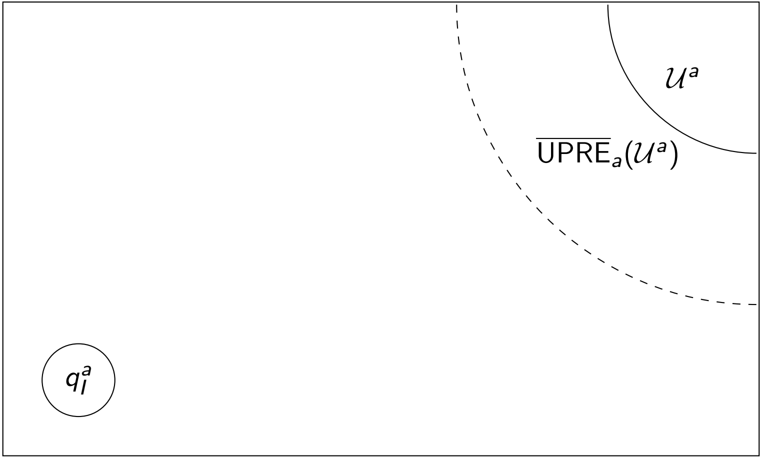
Forward exploration

1. Look for a strategy of environment in the abstract game.



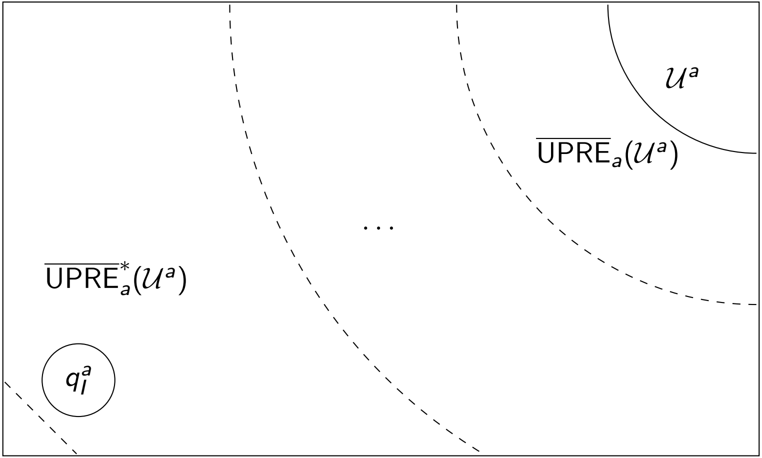
Forward exploration

1. Look for a strategy of environment in the abstract game.



Forward exploration

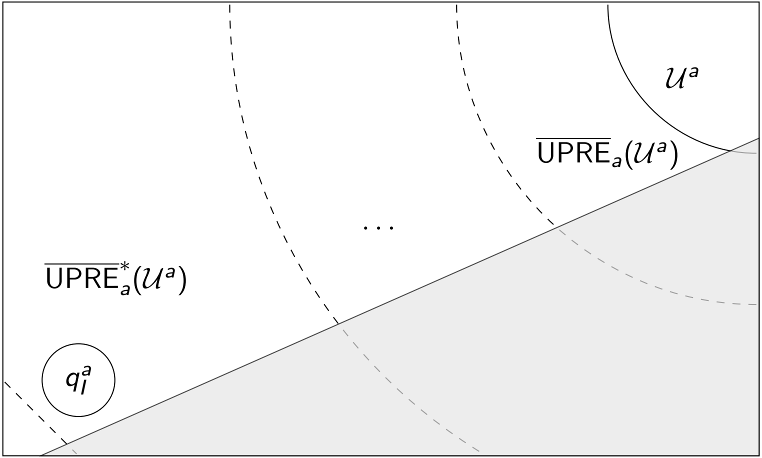
1. Look for a strategy of environment in the abstract game.



$\mu X.(\mathcal{U}^a \cup \overline{\mathsf{UPRE}}_a(X))$

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.



$\mu X.(\mathcal{U}^a \cup \overline{\mathrm{UPRE}}_a(X)) \cap R^a$

Contribution: strategies winning for environment

This allows for two nice optimizations!

Corollary (Over-approx reachable and restrict UPRE)

1. *If $q_I^a \in \overline{UPRE}_a^*(\mathcal{U}^a)$ then we can restrict our search to states reachable if environment plays $\Lambda^a(P, X_u)$,*
2. *and we can replace UPRE by*

$$UPRE_\Lambda(S) = \exists X_u, \forall X_c, \exists L' : T(L, X_u, X_c, L') \wedge \Lambda(L, X_u) \wedge S(L')$$

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 $\mathcal{U}^a$ and current visible latches,

1. we consider a latch l interesting if $l \not\Rightarrow \mathcal{U}^a$ and $\neg l \not\Rightarrow \mathcal{U}^a$; 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 Δ^a more closely resemble the original δ.

Contribution: when is a latch “useful”?

We don’t have a unique answer :-(

Definition (Interesting and useful latches)

Given $\mathcal{U}^a$ and current visible latches,

1. we consider a latch l interesting if $l \not\Rightarrow \mathcal{U}^a$ and $\neg l \not\Rightarrow \mathcal{U}^a$; 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 Δ^a more closely resemble the original δ.

abs_synth(G, G^a, R^a)

$w_u := \mu X.\ (\mathcal{U}^a \cup \underline{\mathrm{UPRE}}_a(X)) \cap R^a$;
if $q_I^a \in w_u$ **then** **return** not controllable;
$prev := \emptyset$;
while $R^a \neq prev$ **do**
    $prev := R^a$;
    $W_u := \mu X.\ (w_u \cup \overline{\mathrm{UPRE}}_a(X)) \cap R^a$;
    if $q_I^a \notin W_u$ **then** **return** controllable;
    Λ^{env} := non-det strategy defined by (w_u);
    $R^a := \mu X.\ (q_I^a \cup \mathrm{post}(X, \Lambda^{env})) \cap R^a$;
end
$w'_u := (\mathrm{UPRE}_{\gamma(\Lambda^{env})}(\gamma(w_u))) \cap \gamma(R^a)$;
if $w'_u \subseteq \gamma(w_u)$ **then** **return** controllable;
$Q_2^a :=$ refine$(Q^a, w'_u \cup \gamma(w_u), \gamma(R^a))$;
$\mathcal{U}_2^a := \underline{\alpha}_2(w'_u \cup \gamma(w_u))$;
return abs_synth$(G, G_2^a, \overline{\alpha}_2(\gamma(R^a)))$;

$\texttt{abs\_synth}(G, G^a, R^a)$

Is q_I^a already in the FP of under-approx'd UPRE?

$w_u := \mu X.\ (\mathcal{U}^a \cup \underline{\mathrm{UPRE}}_a(X)) \cap R^a$;
if $q_I^a \in w_u$ **then** **return** not controllable;
$prev := \emptyset$;
while $R^a \neq prev$ **do**
    $prev := R^a$;
    $W_u := \mu X.\ (w_u \cup \overline{\mathrm{UPRE}}_a(X)) \cap R^a$;
    if $q_I^a \notin W_u$ **then** **return** controllable;
    $\Lambda^{env} :=$ non-det strategy defined by (w_u);
    $R^a := \mu X.\ (q_I^a \cup \mathrm{post}(X, \Lambda^{env})) \cap R^a$;
end
$w_u' := (\mathrm{UPRE}_{\gamma(\Lambda^{env})}(\gamma(w_u))) \cap \gamma(R^a)$;
if $w_u' \subseteq \gamma(w_u)$ **then** **return** controllable;
$Q_2^a := \texttt{refine}(Q^a, w_u' \cup \gamma(w_u), \gamma(R^a))$;
$\mathcal{U}_2^a := \underline{\alpha}_2(w_u' \cup \gamma(w_u))$;
return $\texttt{abs\_synth}(G, G_2^a, \overline{\alpha}_2(\gamma(R^a)))$;

abs_synth(G, G^a, R^a)

```
wu := μX. (U^a ∪ UPRE_a(X)) ∩ R^a
if q_I^a ∈ wu then  return not contr
prev := ∅;
while R^a ≠ prev do
    prev := R^a;
    Wu := μX. (wu ∪ ‾UPRE‾_a(X)) ∩ R^a;
    if q_I^a ∉ Wu then  return controllable;
    Λ^env := non-det strategy defined by (wu);
    R^a := μX. (q_I^a ∪ post(X, Λ^env)) ∩ R^a;
end
w'u := (UPRE_γ(Λ^env)(γ(wu))) ∩ γ(R^a);
if w'u ⊆ γ(wu) then  return controllable;
Q_2^a := refine(Q^a, w'u ∪ γ(wu), γ(R^a));
U_2^a := α_2(w'u ∪ γ(wu));
return abs_synth(G, G_2^a, ‾α‾_2(γ(R^a)));
```

$w_u := \mu X.\ (\mathcal{U}^a \cup \underline{\mathrm{UPRE}}_a(X)) \cap R^a$

if $q_I^a \in w_u$ **then** **return** not contr

$prev := \emptyset$;

while $R^a \neq prev$ **do**

- $prev := R^a$;
- $W_u := \mu X.\ (w_u \cup \overline{\mathrm{UPRE}}_a(X)) \cap R^a$;
- **if** $q_I^a \notin W_u$ **then** **return** controllable;
- Λ^{env} := non-det strategy defined by (w_u);
- $R^a := \mu X.\ (q_I^a \cup \mathrm{post}(X, \Lambda^{env})) \cap R^a$;

end

$w_u' := (\mathrm{UPRE}_{\gamma(\Lambda^{env})}(\gamma(w_u))) \cap \gamma(R^a)$;

if $w_u' \subseteq \gamma(w_u)$ **then** **return** controllable;

$Q_2^a :=$ refine$(Q^a, w_u' \cup \gamma(w_u), \gamma(R^a))$;

$\mathcal{U}_2^a := \underline{\alpha}_2(w_u' \cup \gamma(w_u))$;

return abs_synth$(G, G_2^a, \overline{\alpha}_2(\gamma(R^a)))$;

Is q_I^a not in the FP of over-approx'd UPRE?
If it is, just update reachability information

$\texttt{abs\_synth}(G, G^a, R^a)$

$w_u := \mu X.\ (\mathcal{U}^a \cup \underline{\mathrm{UPRE}}_a(X)) \cap R^a$;
if $q_I^a \in w_u$ **then return** not controllable;
$prev := \emptyset$;
while $R^a \neq prev$ **do**
  $prev := R^a$;
  $W_u := \mu X.\ (w_u \cup \overline{\mathrm{UPRE}}_a(X)) \cap R^a$;
  if $q_I^a \notin W_u$ **then return** controllable;
  Λ^{env} := non-det strategy defined by (w_u);
  $R^a := \mu X.\ (q_I^a \cup \mathrm{post}(X, \Lambda$
end
$w'_u := (\mathrm{UPRE}_{\gamma(\Lambda^{env})}(\gamma(w_u))) \cap \gamma(R^a)$;
if $w'_u \subseteq \gamma(w_u)$ **then return** controllable;
$Q_2^a := \texttt{refine}(Q^a, w'_u \cup \gamma(w_u), \gamma(R^a))$;
$\mathcal{U}_2^a := \underline{\alpha}_2(w'_u \cup \gamma(w_u))$;
return $\texttt{abs\_synth}(G, G_2^a, \overline{\alpha}_2(\gamma(R^a)))$;

Is the under-approx'd UPRE
FP the concrete FP as well?

abs_synth(G, G^a, R^a)

$w_u := \mu X.\ (\mathcal{U}^a \cup \underline{\mathrm{UPRE}}_a(X)) \cap R^a$;
if $q_I^a \in w_u$ **then** **return** not controllable;
$prev := \emptyset$;
while $R^a \neq prev$ **do**
    $prev := R^a$;
    $W_u := \mu X.\ (w_u \cup \overline{\mathrm{UPRE}}_a(X)) \cap R^a$;
    if $q_I^a \notin W_u$ **then** **return** controllable;
    $\Lambda^{env} :=$ non-det strategy defined by (w_u);
    $R^a := \mu X.\ (q_I^a \cup \mathrm{post}(X, \Lambda^{env})) \cap R^a$;
end
$w'_u := (\mathrm{UPRE}_{\gamma(\Lambda^{env})}(\gamma(w_u))) \cap$
if $w'_u \subseteq \gamma(w_u)$ **then** **return** controllable;
$Q_2^a :=$ refine$(Q^a, w'_u \cup \gamma(w_u), \gamma(R^a))$;
$\mathcal{U}_2^a := \underline{\alpha}_2(w'_u \cup \gamma(w_u))$;
return abs_synth$(G, G_2^a, \overline{\alpha}_2(\gamma(R^a)))$;

Add a new latch, update $\mathcal{U}^a$, R^a and repeat.

Some results

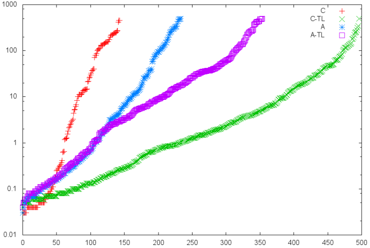


Figure : Time (in seconds) to check realizability.

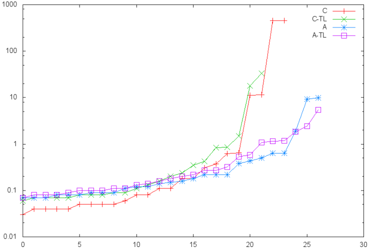


Figure : Time (in seconds) for *cnt* benchmarks.

- (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]

Thank you for your attention!

If you want to ~~drink~~ download our tool:

https://github.com/gaperez64/AbsSynthe

References I

Bloem, R., Könighofer, R., and Seidl, M. (2014).
Sat-based synthesis methods for safety specs.
In *VMCAI*, volume 8318 of *LNCS*, pages 1–20. Springer.

Coudert, O., Berthet, C., and Madre, J. C. (1990).
Verification of synchronous sequential machines based on symbolic execution.
In *Automatic verification methods for finite state systems*, volume 407 of *LNCS*, pages 365–373. Springer.

Coudert, O., Madre, J. C., and Berthet, C. (1991).
Verifying temporal properties of sequential machines without building their state diagrams.
In *CAV*, volume 531 of *LNCS*, pages 23–32. Springer.

References II

de Alfaro, L. and Roy, P. (2010).
Solving games via three-valued abstraction refinement.
Information and Computation, 208(6):666–676.