Solving Games Via Three-Valued Abstraction Refinement

Georg Neis
Advisor: Rayna Dimitrova

July 17, 2008
Games are important for verification and synthesis.

- problem: size of state-space
- solution: abstraction
Games

- game structure $G = (S, \lambda, \delta)$
- turn function $\lambda : S \to \{1, 2\}$ (so $S = S_1 \cup S_2$)
- transition function $\delta : S \to 2^S \setminus \emptyset$
Example

\[ S = \{1, 2, 3, 4, 5, 6, 7\} \]
game objective is an $\omega$-regular language $\Phi \subseteq S^\omega$

to win, sequence of states must be in this language

here: reachability and safety

reachability: $\Diamond \ T$ where $T \subseteq S$ denotes
$\{ \sigma \in S^\omega \mid \exists k \geq 0. \sigma[k] \in T \}$

safety: $\Box \ T$ where $T \subseteq S$ denotes
$\{ \sigma \in S^\omega \mid \forall k \geq 0. \sigma[k] \in T \}$
Strategies

- strategy is a function $\pi_i : S^* \times S_i \rightarrow S$
- outcome($s, \pi_1, \pi_2$) = $\sigma \in S^\omega$ such that $\forall k \geq 0. \sigma[k] \in S_i \implies \sigma[k+1] = \pi_i(\sigma[0..k])$
- $s$ is winning for player 1 with objective $\Phi$ iff $\exists \pi_1. \forall \pi_2. \text{outcome}(s, \pi_1, \pi_2) \in \Phi$
- $\langle 1 \rangle \Phi := \{ s \in S \mid s \text{ is winning for player 1 with objective } \Phi \}$
Controllable Predecessors

\[ \text{cpre}_1 : 2^S \rightarrow 2^S \]
\[ \text{cpre}_1(T) = \{ s \in S_1 \mid \delta(s) \cap T \neq \emptyset \} \cup \{ s \in S_2 \mid \delta(s) \subseteq T \} \]
Goal

- given game objective $\Phi$
- given set of initial states $I \subseteq S$
- decide $I \cap \langle 1 \rangle \Phi \supseteq \emptyset$
_example

\[ \Phi = \diamond \{7\} \]
\[ \text{cpre}_1(\{7\}) = \{5, 6, 7\} \]
\[ \text{cpre}_1(\{5, 6, 7\}) = \{3, 5, 6, 7\} \]
\[ \text{cpre}_1(\{3, 5, 6, 7\}) = \{1, 3, 5, 6, 7\} \]
\[ \langle 1 \rangle \Phi = \{1, 3, 5, 6, 7\} \]
an abstraction of $G = (S, \lambda, \delta)$ is a set $V \subseteq 2^S \setminus \{\emptyset\}$ of abstract states such that $\bigcup V = S$ so each abstract state is a nonempty set of concrete states
Abstractions
$V = \{A, B\} = \{\{1, 2, 3, 4\}, \{5, 6\}\}$
Abstractions

concrete states corresponding to a set $U$ of abstract states:

$$U\downarrow := \bigcup_{u \in U} u$$
Abstractions

concrete states corresponding to a set $U$ of abstract states:

$$U\downarrow := \bigcup_{u \in U} u$$

for instance: $\{B\} \downarrow = \{5, 6, 7\}$, $\{A, B\} \downarrow = S$
abstract states corresponding to a set $T$ of concrete states?
abstract states corresponding to a set $T$ of concrete states?

- *under-approximation* $T_{\text{under}} := \{v \in V \mid v \subseteq T\}$

  e.g. $\{1\}_{\text{under}} = \emptyset$ and $\{1, 3, 5, 6, 7\}_{\text{under}} = \{B\}$
abstract states corresponding to a set $T$ of concrete states?

- **under-approximation** $T_{\text{under}} := \{v \in V \mid v \subseteq T\}$
  e.g. $\{1\}_{\text{under}} = \emptyset$ and $\{1, 3, 5, 6, 7\}_{\text{under}} = \{B\}$

- **over-approximation** $T_{\text{over}} := \{v \in V \mid v \cap T \neq \emptyset\}$
  e.g. $\{1\}_{\text{over}} = \{A\}$ and $\{1, 3, 5, 6, 7\}_{\text{over}} = \{A, B\}$
Abstractions

- for any $T \subseteq S$ we have $T^{\text{under}} \downarrow \subseteq T \subseteq T^{\text{over}} \downarrow$
- abstraction is *precise* for $T$ iff $T^{\text{under}} = T^{\text{over}}$
Abstraction Refinement

- how to find a good abstraction?
- approach: *abstraction refinement*
- popular technique: CEGAR
- alternative proposal: three-valued analysis
Abstraction Refinement

- take abstraction
- compute must-win states, never-win states, and may-win states
- if not sufficiently precise: reduce number of may-win states and repeat
- refinement depends on the property in question!
Abstraction Refinement

- in concrete game: state is winning if it's a cpre of a winning state
- in abstract game? approximate!
Algorithm for Reachability Games

while true do
  \( W_{\text{must}} := \mu Y. (T^{\text{under}} \cup \text{cpre}_1(Y^{\downarrow})^{\text{under}}) \)
  \( W_{\text{may}} := \mu Y. (T^{\text{over}} \cup \text{cpre}_1(Y^{\downarrow})^{\text{over}}) \)
  if \( W_{\text{may}} \cap I^{\text{over}} = \emptyset \) then return NO
  if \( W_{\text{must}} \cap I^{\text{under}} \neq \emptyset \) then return YES
  choose \( v \in (W_{\text{may}} \setminus W_{\text{must}}) \cap \text{cpre}_1(W_{\text{must}}^{\downarrow})^{\text{over}} \)
  let \( v_1 = v \cap \text{cpre}_1(W_{\text{must}}^{\downarrow}) \)
  let \( v_2 = v \setminus v_1 \)
  \( V := (V \setminus \{v\}) \cup \{v_1, v_2\} \)

done
Algorithm for Reachability Games

A

B

C

D

1

2

3

4

5

6

7
Algorithm for Reachability Games

\[ W_{\text{must}} := \mu Y. (T^{\text{under}} \cup \text{cpre}_1(Y_{\downarrow})^{\text{under}}) = \{C, D\} \]

\[ W_{\text{may}} := \mu Y. (T^{\text{over}} \cup \text{cpre}_1(Y_{\downarrow})^{\text{over}}) = \{A, B, C, D\} \]
Algorithm for Reachability Games

if $W_{\text{may}} \cap I^{\text{over}} = \emptyset$ then return NO
if $W_{\text{must}} \cap I^{\text{under}} \neq \emptyset$ then return YES
choose \( v \in (W_{\text{may}} \setminus W_{\text{must}}) \cap \text{cpre}_1(W_{\text{must} \downarrow})^{\text{over}} \)
let \( v_1 = v \cap \text{cpre}_1(\mathcal{W}_{\text{must} \downarrow}) \)
let \( v_2 = v \setminus v_1 \)
Algorithm for Reachability Games

\[ V := (V \setminus \{v\}) \cup \{v_1, v_2\} \]
Algorithm for Safety Games

- dual to reachability: $\langle 1 \rangle □ T = S \setminus \langle 2 \rangle ◊ (S \setminus T)$
Algorithm for Safety Games

- dual to reachability: \( (1) \square T = S \setminus (2) \Diamond (S \setminus T) \)
- refinement for reachability:
  choose \( v \in (W_{\text{may}} \setminus W_{\text{must}}) \cap \text{cpre}_1(W_{\text{must}\downarrow})^{\text{over}} \)
Algorithm for Safety Games

- dual to reachability: \( \langle 1 \rangle □ T = S \setminus \langle 2 \rangle □ (S \setminus T) \)

- refinement for reachability:
  
  choose \( v \in (W_{\text{may}} \setminus W_{\text{must}}) \cap \text{cpre}_1(W_{\text{must}\downarrow})^{\text{over}} \)

- refinement for safety:
  
  choose \( v \in (W_{\text{may}} \setminus W_{\text{must}}) \cap \text{cpre}_2(W_{\text{must}\downarrow})^{\text{over}} \)

  i.e., \( v \in (W_{\text{may}} \setminus W_{\text{must}}) \cap \text{cpre}_2(V \setminus W_{\text{may}\downarrow})^{\text{over}} \)
while true do
    \( W_{\text{must}} := \nu Y. (T^{\text{under}} \cap \text{cpre}_1(Y^{\downarrow})^{\text{under}}) \)
    \( W_{\text{may}} := \nu Y. (T^{\text{over}} \cap \text{cpre}_1(Y^{\downarrow})^{\text{over}}) \)
    if \( W_{\text{may}} \cap I^{\text{over}} = \emptyset \) then return NO
    if \( W_{\text{must}} \cap I^{\text{under}} \neq \emptyset \) then return YES
    choose \( v \in (W_{\text{may}} \setminus W_{\text{must}}) \cap \text{cpre}_2(V \setminus W_{\text{may}}^{\downarrow})^{\text{over}} \)
    let \( v_1 = v \cap \text{cpre}_2(S \setminus W_{\text{may}}^{\downarrow}) \)
    let \( v_2 = v \setminus v_1 \)
    \( V := (V \setminus \{v\}) \cup \{v_1, v_2\} \)

    done
Algorithm for Safety Games

\[ \Phi = \square \{1, 2, 3, 4\} \]
Algorithm for Safety Games

\[ W_{\text{must}} := \nu Y. (T^{\text{under}} \cap \text{cpre}_1(Y\downarrow)^{\text{under}}) = \{C\} \]

\[ W_{\text{may}} := \nu Y. (T^{\text{over}} \cap \text{cpre}_1(Y\downarrow)^{\text{over}}) = \{A, B, C\} \]
if $W_{\text{may}} \cap I^\text{over} = \emptyset$ then return NO
if $W_{\text{must}} \cap I^\text{under} \neq \emptyset$ then return YES
choose $v \in (W_{\text{may}} \setminus W_{\text{must}}) \cap \text{cpre}_2(V \setminus W_{\text{may}\downarrow})^\text{over}$
Algorithm for Safety Games

let $v_1 = v \cap \text{cpre}_2(S \setminus W_{may\downarrow})$

let $v_2 = v \setminus v_1$
Algorithm for Safety Games

\[ V := (V \setminus \{v\}) \cup \{v_1, v_2\} \]
Termination of the Algorithms

- correctness ✓
- termination?
Termination of the Algorithms

- correctness ✓
- termination? at least if there exists a finite region algebra for the game structure, i.e., an abstraction that is
  - closed under boolean operations
  - closed under controllable predecessor operators
Comparison to CEGAR

- 3-valued approach never needs more refinement steps
- however, CEGAR may need more than 3-valued approach
- reason is loss of precision due to abstract edges