Recap: Galois Connections

Outline of Lecture 12

Recap: Galois Connections

Recap: Concrete Semantics of WHILE Programs

Safe Approximation of Functions

Safe Approximation of Execution Relations

Examples
Recap: Galois Connections

Galois Connections

Definition (Galois connection)

Let \((L, \sqsubseteq_L)\) and \((M, \sqsubseteq_M)\) be complete lattices. A pair \((\alpha, \gamma)\) of monotonic functions

\[
\alpha : L \to M \quad \text{and} \quad \gamma : M \to L
\]

is called a Galois connection if

\[
\forall l \in L : l \sqsubseteq_L \gamma(\alpha(l)) \quad \text{and} \quad \forall m \in M : \alpha(\gamma(m)) \sqsubseteq_M m
\]

Interpretation:

- \(L = \{\text{sets of concrete values}\}\), \(M = \{\text{sets of abstract values}\}\)
- \(\alpha = \text{abstraction function, } \gamma = \text{concretisation function}\)
- \(l \sqsubseteq_L \gamma(\alpha(l))\): \(\alpha\) yields over-approximation
- \(\alpha(\gamma(m)) \sqsubseteq_M m\): no loss of precision by abstraction after concretisation
- Usually: \(l \neq \gamma(\alpha(l)), \alpha(\gamma(m)) = m\)
Recap: Galois Connections

Properties of Galois Connections

Lemma

Let \((\alpha, \gamma)\) be a Galois connection with \(\alpha : L \rightarrow M\) and \(\gamma : M \rightarrow L\), and let \(l \in L\), \(m \in M\), \(L' \subseteq L\), \(M' \subseteq M\).

1. \(\alpha(l) \subseteq_M m \iff l \subseteq_L \gamma(m)\)
2. \(\gamma\) is uniquely determined by \(\alpha\) as follows: \(\gamma(m) = \bigsqcup\{l \in L \mid \alpha(l) \subseteq_M m\}\)
3. \(\alpha\) is uniquely determined by \(\gamma\) as follows: \(\alpha(l) = \bigsqcap\{m \in M \mid l \subseteq_L \gamma(m)\}\)
4. \(\alpha\) is completely distributive: for every \(L' \subseteq L\), \(\alpha(\bigsqcup L') = \bigsqcup\{\alpha(l) \mid l \in L'\}\)
5. \(\gamma\) is completely multiplicative: for every \(M' \subseteq M\), \(\gamma(\bigsqcap M') = \bigsqcap\{\gamma(m) \mid m \in M'\}\)

Proof.

on the board
Recap: Concrete Semantics of WHILE Programs

Outline of Lecture 12

Recap: Galois Connections

Recap: Concrete Semantics of WHILE Programs

Safe Approximation of Functions

Safe Approximation of Execution Relations

Examples
Recap: Concrete Semantics of WHILE Programs

Execution of Statements I

**Definition (Execution relation for statements)**

If $c \in \text{Cmd}$ and $\sigma \in \Sigma$, then $\langle c, \sigma \rangle$ is called a configuration. The execution relation

$$\rightarrow \subseteq (\text{Cmd} \times \Sigma) \times ((\text{Cmd} \cup \{\downarrow\}) \times \Sigma)$$

is defined by the following rules:

- **(skip)**
  $$\langle \text{skip}, \sigma \rangle \rightarrow \langle \downarrow, \sigma \rangle$$

- **(asgn)**
  $$\langle x := a, \sigma \rangle \rightarrow \langle \downarrow, \sigma[x \mapsto \text{val}_\sigma(a)] \rangle$$

- **(seq1)**
  $$\langle c_1, \sigma \rangle \rightarrow \langle c'_1, \sigma' \rangle \quad c'_1 \neq \downarrow$$
  $$\langle c_1; c_2, \sigma \rangle \rightarrow \langle c'_1; c_2, \sigma' \rangle$$

- **(seq2)**
  $$\langle c_1, \sigma \rangle \rightarrow \langle \downarrow, \sigma' \rangle$$
  $$\langle c_1; c_2, \sigma \rangle \rightarrow \langle c_2, \sigma' \rangle$$
Recap: Concrete Semantics of WHILE Programs

Execution of Statements II

Definition (Execution relation for statements; continued)

<table>
<thead>
<tr>
<th>Condition</th>
<th>Expression</th>
<th>Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(if1)</td>
<td>$val_\sigma(b) = \text{true}$</td>
<td>$\langle \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end, } \sigma \rangle \rightarrow \langle c_1, \sigma \rangle$</td>
</tr>
<tr>
<td>(if2)</td>
<td>$val_\sigma(b) = \text{false}$</td>
<td>$\langle \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end, } \sigma \rangle \rightarrow \langle c_2, \sigma \rangle$</td>
</tr>
<tr>
<td>(wh1)</td>
<td>$val_\sigma(b) = \text{true}$</td>
<td>$\langle \text{while } b \text{ do } c \text{ end, } \sigma \rangle \rightarrow \langle c; \text{while } b \text{ do } c \text{ end, } \sigma \rangle$</td>
</tr>
<tr>
<td>(wh2)</td>
<td>$val_\sigma(b) = \text{false}$</td>
<td>$\langle \text{while } b \text{ do } c \text{ end, } \sigma \rangle \rightarrow \langle \downarrow, \sigma \rangle$</td>
</tr>
</tbody>
</table>

Remark: $\downarrow$ indicates successful termination of the program.
Recap: Concrete Semantics of WHILE Programs

Determinism Property of Execution Relation

This operational semantics is well defined in the following sense:

**Theorem**

The execution relation for statements is **deterministic**, i.e., whenever \( c \in \text{Cmd}, \sigma \in \Sigma \) and \( \kappa_1, \kappa_2 \in (\text{Cmd} \cup \{\downarrow\}) \times \Sigma \) such that \( \langle c, \sigma \rangle \rightarrow \kappa_1 \) and \( \langle c, \sigma \rangle \rightarrow \kappa_2 \), then \( \kappa_1 = \kappa_2 \).

**Proof.**

omitted
Safe Approximation of Functions

Outline of Lecture 12

Recap: Galois Connections

Recap: Concrete Semantics of WHILE Programs

Safe Approximation of Functions

Safe Approximation of Execution Relations

Examples
Safe Approximation of Functions

Safe Approximation of Functions I

Definition 12.1 (Safe approximation)

Let \((\alpha, \gamma)\) be a Galois connection with \(\alpha : L \rightarrow M\) and \(\gamma : M \rightarrow L\), and let \(f : L^n \rightarrow L\) and \(f^\# : M^n \rightarrow M\) be functions of rank \(n \in \mathbb{N}\). Then \(f^\#\) is called a safe approximation of \(f\) if, whenever \(m_1, \ldots, m_n \in M\),

\[
\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \subseteq_M f^\#(m_1, \ldots, m_n).
\]

Moreover, \(f^\#\) is called most precise if the reverse inclusion is also true.

Interpretation:
- the abstraction \(f^\#\) covers all concrete \(f\)-results
- Note: monotonicity of \(f\) and/or \(f^\#\) is not required (but usually given; see Lemma 12.3)

Abstract

\[
\begin{array}{c}
\vec{m} \\
\downarrow f^\# \\
f^\#(\vec{m}) \sqsubseteq \alpha(f(\gamma(\vec{m})))
\end{array}
\]

Concrete

\[
\begin{array}{c}
\gamma(\vec{m}) \\
\downarrow f \\
f(\gamma(\vec{m}))
\end{array}
\]
Safe Approximation of Functions

Safe Approximation of Functions I

Definition 12.1 (Safe approximation)

Let \((\alpha, \gamma)\) be a Galois connection with \(\alpha : L \rightarrow M\) and \(\gamma : M \rightarrow L\), and let \(f : L^n \rightarrow L\) and \(f^\# : M^n \rightarrow M\) be functions of rank \(n \in \mathbb{N}\). Then \(f^\#\) is called a safe approximation of \(f\) if, whenever \(m_1, \ldots, m_n \in M\),

\[
\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \subseteq_M f^\#(m_1, \ldots, m_n).
\]

Moreover, \(f^\#\) is called most precise if the reverse inclusion is also true.

Abstract

\[
\begin{align*}
\vec{m} \downarrow f^\# & \longrightarrow \gamma(\vec{m}) \\
f^\#(\vec{m}) \supseteq \alpha(f(\gamma(\vec{m}))) & \longleftarrow f(\gamma(\vec{m})).
\end{align*}
\]

Concrete

- **Interpretation:** the abstraction \(f^\#\) covers all concrete \(f\)-results
- **Note:** monotonicity of \(f\) and/or \(f^\#\) is not required (but usually given; see Lemma 12.3)
Example 12.2 (Safeness: $\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \sqsubseteq_M f^#(m_1, \ldots, m_n)$)

1. Parity abstraction (cf. Example 11.2): $L = (2^{\mathbb{Z}}, \subseteq), M = (2^{\{\text{even, odd}\}}, \subseteq)$

   - $n = 0$: for $f = \text{one} \subseteq 2^{\mathbb{Z}} : () \mapsto \{1\}$,
     - $\text{one}^#() = \{\text{odd}\}$ is most precise: $\alpha(\{1\}) = \{\text{odd}\} = \text{one}^#()$
     - $\text{one}^#() = \{\text{even, odd}\}$ is (only) safe: $\alpha(\{1\}) = \{\text{odd}\} \subsetneq \{\text{even, odd}\} = \text{one}^#()$
     - $\text{one}^#() = \{\text{even}\}$ is unsafe: $\alpha(\{1\}) = \{\text{odd}\} \nsubseteq \{\text{even}\} = \text{one}^#()$
Safe Approximation of Functions

Safe Approximation of Functions II

Example 12.2 (Safeness: $\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \sqsubseteq_M f^#(m_1, \ldots, m_n)$)

1. Parity abstraction (cf. Example 11.2): $L = (2^\mathbb{Z}, \subseteq), M = (2^{\{\text{even, odd}\}}, \subseteq)$

   - $n = 0$: for $f = \text{one} \subseteq 2^\mathbb{Z} : (\cdot) \mapsto \{1\}$,
     - $\text{one}^#() = \{\text{odd}\}$ is most precise: $\alpha(\{1\}) = \{\text{odd}\} = \text{one}^#()$
     - $\text{one}^#() = \{\text{even, odd}\}$ is (only) safe: $\alpha(\{1\}) = \{\text{odd}\} \subsetneq \{\text{even, odd}\} = \text{one}^#()$
     - $\text{one}^#() = \{\text{even}\}$ is unsafe: $\alpha(\{1\}) = \{\text{odd}\} \not\subset \{\text{even}\} = \text{one}^#()$

   - $n = 1$: for $f = \text{dec} : 2^\mathbb{Z} \rightarrow 2^\mathbb{Z} : Z \mapsto \{z - 1 \mid z \in Z\}$,
     - $\text{dec}^#(\{\text{even}\}) = \{\text{odd}\}$ is most precise: $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} = \text{dec}^#(\{\text{even}\})$
     - $\text{dec}^#(\{\text{even}\}) = \{\text{odd, even}\}$ is (only) safe:
       - $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} \subset \{\text{odd, even}\} = \text{dec}^#(\{\text{even}\})$
     - $\text{dec}^#(\{\text{even}\}) = \emptyset$ is unsafe: $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} \not\subset \emptyset = \text{dec}^#(\{\text{even}\})$
Safe Approximation of Functions

Safe Approximation of Functions II

Example 12.2 (Safeness: $\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \sqsubseteq_M f^#(m_1, \ldots, m_n)$)

1. Parity abstraction (cf. Example 11.2): $L = (2^Z, \subseteq), M = (2^{\{\text{even}, \text{odd}\}}, \subseteq)$
   - $n = 0$: for $f = \text{one} \subseteq 2^Z : () \mapsto \{1\}$,
     - $\text{one}^#() = \{\text{odd}\}$ is most precise: $\alpha(\{1\}) = \{\text{odd}\} = \text{one}^#()$
     - $\text{one}^#() = \{\text{even}, \text{odd}\}$ is (only) safe: $\alpha(\{1\}) = \{\text{odd}\} \subsetneq \{\text{even}, \text{odd}\} = \text{one}^#()$
     - $\text{one}^#() = \{\text{even}\}$ is unsafe: $\alpha(\{1\}) = \{\text{odd}\} \subsetneq \{\text{even}\} = \text{one}^#()$
   - $n = 1$: for $f = \text{dec} : 2^Z \rightarrow 2^Z : Z \mapsto \{z - 1 \mid z \in Z\}$,
     - $\text{dec}^#(\{\text{even}\}) = \{\text{odd}\}$ is most precise: $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} = \text{dec}^#(\{\text{even}\})$
     - $\text{dec}^#(\{\text{even}\}) = \{\text{odd}, \text{even}\}$ is (only) safe:
       $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} \subsetneq \{\text{odd}, \text{even}\} = \text{dec}^#(\{\text{even}\})$
     - $\text{dec}^#(\{\text{even}\}) = \emptyset$ is unsafe: $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} \not\subseteq \emptyset = \text{dec}^#(\{\text{even}\})$
   - $n = 2$: for $f = + : 2^Z \times 2^Z \rightarrow 2^Z : (z_1, z_2) \mapsto z_1 + z_2$,
     - $\{\text{even}\} +^# \{\text{odd}\} = \{\text{odd}\}$ is m.p.: $\alpha(\gamma(\{\text{even}\}) + \gamma(\{\text{odd}\})) = \{\text{odd}\} = \{\text{even}\} +^# \{\text{odd}\}$
     - $\{\text{even}\} +^# \{\text{odd}\} = \{\text{even}, \text{odd}\}$ is (only) safe:
       $\alpha(\gamma(\{\text{even}\}) + \gamma(\{\text{odd}\})) = \{\text{odd}\} \subsetneq \{\text{even}, \text{odd}\} = \{\text{even}\} +^# \{\text{odd}\}$
     - $\{\text{even}\} +^# \{\text{odd}\} = \{\text{even}\}$ is unsafe:
       $\alpha(\gamma(\{\text{even}\}) + \gamma(\{\text{odd}\})) = \{\text{odd}\} \not\subseteq \{\text{even}\} = \{\text{even}\} +^# \{\text{odd}\}$
Safe Approximation of Functions

Safe Approximation of Functions III

Reminder: \( \alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \subseteq_M f^\#(m_1, \ldots, m_n) \)

Example 12.2 (continued)

Most precise approximations (with \( L = (2^\mathbb{Z}, \subseteq) \)):

2. Sign abstraction (cf. Example 11.3): \( M = (2^{\{+, -, 0\}}, \subseteq) \)
   - \( n = 0 \): \( \text{one}^\#() = \{+\} \)
   - \( n = 1 \): \( \text{dec}^\#(\{+\}) = \{+, 0\}, \quad \text{neg}^\#(\{+\}) = \{-\} \)
   - \( n = 2 \): \( \{+\} +^\# \{+\} = \{+\}, \quad \{+\} -^\# \{+\} = \{+, -, 0\}, \quad \{+\} \cdot^\# \{-\} = \{-\} \)
Safe Approximation of Functions

Safe Approximation of Functions III

Reminder: \( \alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \subseteq_M f^#(m_1, \ldots, m_n) \)

Example 12.2 (continued)

Most precise approximations (with \( L = (2^\mathbb{Z}, \subseteq) \)):

2. Sign abstraction (cf. Example 11.3): \( M = (2^{\{+,-,0\}}, \subseteq) \)
   - \( n = 0 \): one\(^\#\)(\( ) = \{+\}
   - \( n = 1 \): dec\(^\#\)(\( \{+\} \)) = \{+, 0\}, \quad -\#(\{+\}) = \{-\}
   - \( n = 2 \): \( \{+\} +\# \{+\} = \{+\}, \quad \{+\} -\# \{+\} = \{+, -, 0\}, \quad \{+\} \cdot\# \{-\} = \{-\}

3. Interval abstraction (cf. Example 11.4): \( M = ((\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{+\infty\}) \cup \{\emptyset\}, \subseteq) \)
   - \( n = 0 \): one\(^\#\)(\( ) = [1, 1]
   - \( n = 1 \): dec\(^\#\)(\( [z_1, z_2] \)) = [z_1 - 1, z_2 - 1], \quad -\#([z_1, z_2]) = [-z_2, -z_1]
   - \( n = 2 \): \( [y_1, y_2] +\# [z_1, z_2] = [y_1 + z_1, y_2 + z_2] \)
     \( [y_1, y_2] -\# [z_1, z_2] = [y_1 - z_2, y_2 - z_1] \)
     \( [y_1, y_2] \cdot\# [z_1, z_2] = \{y_1z_1, y_1z_2, y_2z_1, y_2z_2\} \)

(\( \text{thus,} \quad +\#/ -\#/ \cdot\# = \oplus/\ominus/\odot \text{ from Slide 7.20} \))
Lemma 12.3

If $f : L^n \to L$ and $f^\# : M^n \to M$ are monotonic, then $f^\#$ is a safe approximation of $f$ iff, for all $l_1, \ldots, l_n \in L$,

$$\alpha(f(l_1, \ldots, l_n)) \sqsubseteq_M f^\#(\alpha(l_1), \ldots, \alpha(l_n)).$$
Lemma 12.3

If \( f : L^n \to L \) and \( f^\# : M^n \to M \) are monotonic, then \( f^\# \) is a safe approximation of \( f \) iff, for all \( l_1, \ldots, l_n \in L \),

\[
\alpha(f(l_1, \ldots, l_n)) \sqsubseteq_M f^\#(\alpha(l_1), \ldots, \alpha(l_n)).
\]

Proof.

on the board
Safe Approximation of Execution Relations

Outline of Lecture 12

Recap: Galois Connections

Recap: Concrete Semantics of WHILE Programs

Safe Approximation of Functions

Safe Approximation of Execution Relations

Examples
Safe Approximation of Execution Relations

Encoding Execution Relations by Transition Functions I

• Reminder: concrete semantics of WHILE
  – statements \( \text{skip} \mid x := a \mid c_1 ; c_2 \mid \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end} \mid \text{while } b \text{ do } c \text{ end} \in \text{Cmd} \)
  – states \( \Sigma := \{ \sigma \mid \sigma : \text{Var} \rightarrow \mathbb{Z} \} \) (Definition 11.6)
  – execution relation \( \rightarrow \subseteq (\text{Cmd} \times \Sigma) \times ((\text{Cmd} \cup \{ \uparrow \}) \times \Sigma) \) (Definition 11.9)
Safe Approximation of Execution Relations

Encoding Execution Relations by Transition Functions I

• Reminder: concrete semantics of WHILE
  – statements \( \text{skip} \mid x := a \mid c_1 ; c_2 \mid \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end} \mid \text{while } b \text{ do } c \text{ end} \in \text{Cmd} \)
  – states \( \Sigma := \{ \sigma \mid \sigma : \text{Var} \rightarrow \mathbb{Z} \} \) (Definition 11.6)
  – execution relation \( \rightarrow \subseteq (\text{Cmd} \times \Sigma) \times ((\text{Cmd} \cup \{ \downarrow \}) \times \Sigma) \) (Definition 11.9)
• Yields concrete domain \( L := (2^{\Sigma}, \subseteq) \) and concrete transition function:

\[
\text{next}_{c,c'} : 2^\Sigma \rightarrow 2^\Sigma
\]

\( c \in \text{Cmd}, c' \in \text{Cmd} \cup \{ \downarrow \} \) and, for every \( S \subseteq \Sigma \),

\[
\text{next}_{c,c'}(S) := \{ \sigma' \in \Sigma \mid \exists \sigma \in S : \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle \}.
\]
Safe Approximation of Execution Relations

Encoding Execution Relations by Transition Functions II

Remarks: \texttt{next} satisfies the following properties

- “Determinism” (cf. Theorem 11.11):
  - for all $c \in \text{Cmd}$, $c' \in \text{Cmd} \cup \{\downarrow\}$ and $\sigma \in \Sigma$, $|\text{next}_{c,c'}(\{\sigma\})| \leq 1$
  - for all $c \in \text{Cmd}$ and $\sigma \in \Sigma$ there exists exactly one $c' \in \text{Cmd} \cup \{\downarrow\}$ such that $\text{next}_{c,c'}(\{\sigma\}) \neq \emptyset$

When is $\text{next}_{c,c'}(S) = \emptyset$? Possible reasons:

1. $S = \emptyset$
2. $c'$ is not a possible successor statement of $c$, e.g., $\square c = (x := 0)$
3. $c'$ is unreachable for all $\sigma \in S$, e.g., $\square c = (\text{if } x = 0 \text{ then } x := 1 \text{ else skip end})$

$\square \sigma(x) = 0$ for each $\sigma \in S$
Remarks: \texttt{next} satisfies the following properties

- **“Determinism”** (cf. Theorem 11.11):
  - for all \( c \in \text{Cmd}, c' \in \text{Cmd} \cup \{\downarrow\} \) and \( \sigma \in \Sigma \), \( |\text{next}_{c,c'}(\sigma)| \leq 1 \)
  - for all \( c \in \text{Cmd} \) and \( \sigma \in \Sigma \) there exists exactly one \( c' \in \text{Cmd} \cup \{\downarrow\} \) such that \( \text{next}_{c,c'}(\{\sigma\}) \neq \emptyset \)

- When is \( \text{next}_{c,c'}(S) = \emptyset \)? Possible reasons:
  1. \( S = \emptyset \)
  2. \( c' \) is not a possible successor statement of \( c \), e.g.,
     - \( c = (x := 0) \)
     - \( c' = \text{skip} \)
  3. \( c' \) is unreachable for all \( \sigma \in S \), e.g.,
     - \( c = (\text{if } x = 0 \text{ then } x := 1 \text{ else } \text{skip end}) \)
     - \( c' = \text{skip} \)
     - \( \sigma(x) = 0 \) for each \( \sigma \in S \)
Safe Approximation of Execution Relations

Reminder: abstraction determined by Galois connection \((\alpha, \gamma)\) with \(\alpha : L \rightarrow M\), \(\gamma : M \rightarrow L\)

- here: \(L := 2^\Sigma\), \(M\) not fixed
- usually \(M = Var \rightarrow \ldots\) (more efficient) or \(M = 2^{\text{Var} \rightarrow \ldots}\) (more precise)
- write \(Abs\) in place of \(M\)
- thus \(\alpha : 2^\Sigma \rightarrow Abs\) and \(\gamma : Abs \rightarrow 2^\Sigma\)
Safe Approximation of Execution Relations

Reminder: abstraction determined by Galois connection \((\alpha, \gamma)\) with \(\alpha : L \rightarrow M\), \(\gamma : M \rightarrow L\)

- here: \(L := 2^\Sigma\), \(M\) not fixed
- usually \(M = \text{Var} \rightarrow \ldots\) (more efficient) or \(M = 2^{\text{Var} \rightarrow \ldots}\) (more precise)
- write \(\text{Abs}\) in place of \(M\)
- thus \(\alpha : 2^\Sigma \rightarrow \text{Abs}\) and \(\gamma : \text{Abs} \rightarrow 2^\Sigma\)

Definition 12.5 (Abstract semantics of WHILE)

Given \(\alpha : 2^\Sigma \rightarrow \text{Abs}\), an abstract semantics is defined by a family of functions

\[\text{next}^\#_{c,c'} : \text{Abs} \rightarrow \text{Abs}\]

where \(c \in \text{Cmd}\), \(c' \in \text{Cmd} \cup \{\downarrow\}\), and each \(\text{next}^\#_{c,c'}\) is a safe approximation of \(\text{next}_{c,c'}\), i.e.,

\[\alpha\left(\text{next}^\#_{c,c'}(\gamma(\text{abs}))\right) \sqsubseteq_{\text{Abs}} \text{next}^\#_{c,c'}(\text{abs})\]

for every \(\text{abs} \in \text{Abs}\) (notation: \(\langle c, \text{abs} \rangle \Rightarrow \langle c', \text{abs}' \rangle\) for \(\text{next}^\#_{c,c'}(\text{abs}) = \text{abs}'\)).
Examples

Outline of Lecture 12

Recap: Galois Connections

Recap: Concrete Semantics of WHILE Programs

Safe Approximation of Functions

Safe Approximation of Execution Relations

Examples
Example: Parity Abstraction

Example 12.6 (Parity abstraction (cf. Example 11.2))

- $Var = \{ n \}$
- $Abs = 2^{Var \rightarrow \{ \text{even, odd} \}}$
- Notation: $[n \mapsto p] \in abs \in Abs$ for $p \in \{ \text{even, odd} \}$
Example: Parity Abstraction

Example 12.6 (Parity abstraction (cf. Example 11.2))

- $Var = \{n\}$
- $Abs = 2^{Var \rightarrow \{\text{even, odd}\}}$
- Notation: $[n \mapsto p] \in abs \in Abs$ for $p \in \{\text{even, odd}\}$
- Some abstract transitions:

\[
\langle n := 3 \times n + 1, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \langle \downarrow, \{[n \mapsto \text{even}]\} \rangle
\]
Example: Parity Abstraction

Example 12.6 (Parity abstraction (cf. Example 11.2))

- \( Var = \{n\} \)
- \( Abs = 2^{Var \rightarrow \{\text{even, odd}\}} \)
- Notation: \([n \mapsto p] \in abs \in Abs\) for \(p \in \{\text{even, odd}\}\)
- Some abstract transitions:
  \[
  \langle n := 3 \ast n + 1, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \langle \downarrow, \{[n \mapsto \text{even}]\} \rangle
  \]
  \[
  \langle n := 2 \ast n + 1, \{[n \mapsto \text{even}], [n \mapsto \text{odd}]\} \rangle \Rightarrow \langle \downarrow, \{[n \mapsto \text{odd}]\} \rangle
  \]
Examples

Example: Parity Abstraction

Example 12.6 (Parity abstraction (cf. Example 11.2))

- \(\text{Var} = \{n\}\)
- \(\text{Abs} = 2^{\text{Var} \rightarrow \{\text{even}, \text{odd}\}}\)
- Notation: \([n \mapsto p] \in \text{abs} \in \text{Abs}\) for \(p \in \{\text{even}, \text{odd}\}\)
- Some abstract transitions:
  
  \[\langle n := 3 \ast n + 1, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \downarrow, \{[n \mapsto \text{even}]\}\]
  
  \[\langle n := 2 \ast n + 1, \{[n \mapsto \text{even}], [n \mapsto \text{odd}]\} \rangle \Rightarrow \downarrow, \{[n \mapsto \text{odd}]\}\]
  
  \[\langle \text{while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \downarrow, \{[n \mapsto \text{odd}]\}\]
Example: Parity Abstraction

Example 12.6 (Parity abstraction (cf. Example 11.2))

- \( Var = \{n\} \)
- \( Abs = 2^{\mathit{Var} \rightarrow \{\text{even}, \text{odd}\}} \)
- Notation: \([n \mapsto p] \in abs \in Abs\) for \(p \in \{\text{even}, \text{odd}\}\)
- Some abstract transitions:

\[
\langle n := 3 \ast n + 1, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \langle \bot, \{[n \mapsto \text{even}]\} \rangle
\]
\[
\langle n := 2 \ast n + 1, \{[n \mapsto \text{even}], [n \mapsto \text{odd}]\} \rangle \Rightarrow \langle \bot, \{[n \mapsto \text{odd}]\} \rangle
\]
\[
\langle \text{while } \neg(n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \langle \bot, \{[n \mapsto \text{odd}]\} \rangle
\]
\[
\langle \text{while } \neg(n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \langle c; \text{ while } \neg(n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{odd}]\} \rangle
\]
Example: Parity Abstraction

Example 12.6 (Parity abstraction (cf. Example 11.2))

- \( \text{Var} = \{n\} \)
- \( \text{Abs} = 2^{\text{Var}} \rightarrow \{\text{even, odd}\} \)
- Notation: \([n \mapsto p] \in \text{abs} \in \text{Abs}\) for \(p \in \{\text{even, odd}\}\)
- Some abstract transitions:

\[
\langle n := 3 \times n + 1, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \langle \downarrow, \{[n \mapsto \text{even}]\} \rangle
\]

\[
\langle n := 2 \times n + 1, \{[n \mapsto \text{even}], [n \mapsto \text{odd}]\} \rangle \Rightarrow \langle \downarrow, \{[n \mapsto \text{odd}]\} \rangle
\]

\[
\langle \text{while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \langle \downarrow, \{[n \mapsto \text{odd}]\} \rangle
\]

\[
\langle \text{while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \langle c; \text{ while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{odd}]\} \rangle
\]

\[
\langle \text{while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{even}]\} \rangle \Rightarrow \langle \downarrow, \emptyset \rangle
\]
Example: Parity Abstraction

Example 12.6 (Parity abstraction (cf. Example 11.2))

- $\text{Var} = \{n\}$
- $\text{Abs} = 2^{\text{Var} \rightarrow \{\text{even, odd}\}}$
- Notation: $[n \mapsto p] \in \text{abs} \in \text{Abs}$ for $p \in \{\text{even, odd}\}$
- Some abstract transitions:
  \[
  \langle n := 3 \times n + 1, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \downarrow, \{[n \mapsto \text{even}]\}
  \]
  \[
  \langle n := 2 \times n + 1, \{[n \mapsto \text{even}], [n \mapsto \text{odd}]\} \rangle \Rightarrow \downarrow, \{[n \mapsto \text{odd}]\}
  \]
  \[
  \langle \text{while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \downarrow, \{[n \mapsto \text{odd}]\}
  \]
  \[
  \langle \text{while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{odd}]\} \rangle \Rightarrow \langle c; \text{ while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{odd}]\} \rangle
  \]
  \[
  \langle \text{while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{even}]\} \rangle \Rightarrow \downarrow, \emptyset
  \]
  \[
  \langle \text{while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{even}]\} \rangle \Rightarrow \langle c; \text{ while } \neg (n=1) \text{ do } c \text{ end}, \{[n \mapsto \text{even}]\} \rangle
  \]
Examples

Example: Hailstone Sequences

Example 12.7 (Hailstone Sequences)

```plaintext
[skip]¹;
while [¬(n = 1)]² do
  if [even(n)]³ then
    [n := n / 2]⁴;[skip]⁵
  else
    [n := 3 * n + 1]⁶;[skip]⁷
  end
end
```

- skip statements only for labels
- abstract transition system for \( \sigma(n) \in \mathbb{Z}_{odd} \): on the board
- formal derivation later
Examples

Example: Hailstone Sequences

Example 12.7 (Hailstone Sequences)

```plaintext
[skip];
while [¬(n = 1)] do
  if [even(n)] then
    [n := n / 2];[skip]
  else
    [n := 3 * n + 1];[skip]
end
```

- skip statements only for labels
- abstract transition system for \( \sigma(n) \in \mathbb{Z}_{\text{odd}} \): on the board
- formal derivation later

- **Collatz Conjecture**: given any \( n > 0 \), the program finally returns 1 (that is, every Hailstone Sequence terminates)
- aka \( 3n + 1 \) Conjecture, Ulam Conjecture, Kakutani's Problem, Thwaites' Conjecture, Hasse's Algorithm, or Syracuse Problem