

# Static Program Analysis

## Lecture 13: Abstract Interpretation III (Abstract Interpretation of WHILE Programs)

Thomas Noll

Lehrstuhl für Informatik 2  
(Software Modeling and Verification)



[noll@cs.rwth-aachen.de](mailto:noll@cs.rwth-aachen.de)

<http://moves.rwth-aachen.de/teaching/ws-1415/spa/>

Winter Semester 2014/15

- 1 Recap: Safe Approximation of Functions and Relations
- 2 Example: Hailstone Sequences
- 3 Abstract Interpretation of WHILE Programs
- 4 Abstract Semantics of WHILE

# Safe Approximation of Functions

## Definition (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  $m_1, \dots, m_n \in M$ ,

$$\alpha(f(\gamma(m_1), \dots, \gamma(m_n))) \sqsubseteq_M f^\#(m_1, \dots, m_n).$$

Moreover it is called **most precise** safe approximation if the reverse inclusion is also true.

Abstract		Concrete
$\vec{m}$	$\xrightarrow{\gamma}$	$\gamma(\vec{m})$
$\downarrow f^\#$		$\downarrow f$
$f^\#(\vec{m}) \supseteq \alpha(f(\gamma(\vec{m})))$	$\xleftarrow{\alpha}$	$f(\gamma(\vec{m}))$

- **Interpretation:** the abstraction  $f^\#$  of  $f$  covers all concrete results
- **Note:** monotonicity of  $f$  and/or  $f^\#$  is *not* required (but usually given; see Lemma 12.5)

# Safe Approximation of Execution Relation I

- **Reminder:** **concrete semantics** of WHILE
  - **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**  $L := 2^\Sigma$  and concrete transition function:

## Definition (Concrete transition function)

The **concrete transition function** of WHILE is defined by the family of functions

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

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

$$\text{next}_{c,c'}(S) := \{\sigma' \in \Sigma \mid \exists \sigma \in S : \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle\}.$$

**Remarks:**  $\text{next}$  satisfies the following properties

- “Determinism” (cf. Theorem 12.2):
  - 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  $c'$  not a possible successor statement of  $c$ , e.g.,
    - $c = (x := 0)$
    - $c' = \text{skip}$
  - 3  $c'$  unreachable for all  $\sigma \in S$ , e.g.,
    - $c = (\text{if } x = 0 \text{ then } x := 1 \text{ else skip})$
    - $c' = \text{skip}$
    - $\sigma(x) = 0$  for each  $\sigma \in S$

# Safe Approximation of Execution Relation III

- **Reminder:** abstraction determined by **Galois connection**  $(\alpha, \gamma)$  with  $\alpha : L \rightarrow M$  and  $\gamma : M \rightarrow L$ 
  - here:  $L := 2^\Sigma$ ,  $M$  not fixed (usually  $M = \text{Var} \rightarrow \dots$  or  $M = 2^{\text{Var} \rightarrow \dots}$ )
  - write  $Abs$  in place of  $M$
  - thus  $\alpha : 2^\Sigma \rightarrow Abs$  and  $\gamma : Abs \rightarrow 2^\Sigma$
- Yields abstract semantics:

## Definition (Abstract semantics of WHILE)

Given  $\alpha : 2^\Sigma \rightarrow Abs$ , an **abstract semantics** is defined by a family of functions

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

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.,

$$\alpha(\text{next}_{c,c'}(\gamma(abs))) \sqsubseteq_{Abs} \text{next}_{c,c'}^\#(abs)$$

for every  $abs \in Abs$ .

Notation:  $\langle c, abs \rangle \Rightarrow \langle c', abs' \rangle$  for  $\text{next}_{c,c'}^\#(abs) = abs'$ .

- 1 Recap: Safe Approximation of Functions and Relations
- 2 Example: Hailstone Sequences
- 3 Abstract Interpretation of WHILE Programs
- 4 Abstract Semantics of WHILE

## Example 13.1 (Hailstone Sequences)

```
[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;
```

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



# Example: Hailstone Sequences

## Example 13.1 (Hailstone Sequences)

```
[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;
```

- additional **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 with 1)
- see [http://en.wikipedia.org/wiki/Collatz\\_conjecture](http://en.wikipedia.org/wiki/Collatz_conjecture)
- AKA  $3n + 1$  Conjecture, Ulam Conjecture, Kakutani's Problem, Thwaites' Conjecture, Hasse's Algorithm, or Syracuse Problem
- New proof attempt by Gerhard Opfer from Hamburg University (<http://preprint.math.uni-hamburg.de/public/papers/hbam/hbam2011-09.pdf>)

- 1 Recap: Safe Approximation of Functions and Relations
- 2 Example: Hailstone Sequences
- 3 Abstract Interpretation of WHILE Programs**
- 4 Abstract Semantics of WHILE

- **Problem:** most precise safe approximation not always definable

## Example 13.2 (Fermat's Last Theorem)

Sign abstraction (cf. Example 11.3) on

```
⟨if  $n > 2 \wedge x^n + y^n = z^n$  then  $n := 1$  else  $n := -1$ ,  $\{[n, x, y, z \mapsto +]\}$ ⟩
```

- **Problem:** most precise safe approximation not always definable

## Example 13.2 (Fermat's Last Theorem)

Sign abstraction (cf. Example 11.3) on

$\langle \text{if } n > 2 \wedge x^n + y^n = z^n \text{ then } n := 1 \text{ else } n := -1, \{[n, x, y, z \mapsto +]\} \rangle$

- Result  $n = 1$  possible iff there exist  $n > 2$  and  $x, y, z \geq 1$  such that  $x^n + y^n = z^n$
- **Fermat's Last Theorem:** equation not solvable
- Final proof by Andrew Wiles and Richard Taylor in 1995

- **Problem:** most precise safe approximation not always definable

## Example 13.2 (Fermat's Last Theorem)

Sign abstraction (cf. Example 11.3) on

```
⟨if n>2 ∧ x^n+y^n=z^n then n:=1 else n:=-1, {[n, x, y, z ↦ +]}⟩
```

- Result  $n = 1$  possible iff there exist  $n > 2$  and  $x, y, z \geq 1$  such that  $x^n + y^n = z^n$
- **Fermat's Last Theorem:** equation not solvable
- Final proof by Andrew Wiles and Richard Taylor in 1995

- More general: solvability of Diophantic equations undecidable
- Thus: resort to **possibly imprecise** safe approximations

# Extraction Functions

- **Assumption:** abstraction determined by pointwise mapping of concrete elements
- If  $L = 2^C$  and  $M = 2^A$  with  $\sqsubseteq_L = \sqsubseteq_M = \subseteq$ , then  $\beta : C \rightarrow A$  is called an **extraction function**
- $\beta$  determines **Galois connection**  $(\alpha, \gamma)$  where

$$\alpha : L \rightarrow M : I \mapsto \beta(I) (= \{\beta(c) \mid c \in I\})$$

$$\gamma : M \rightarrow L : m \mapsto \beta^{-1}(m) (= \{c \in C \mid \beta(c) \in m\})$$

# Extraction Functions

- **Assumption:** abstraction determined by pointwise mapping of concrete elements
- If  $L = 2^C$  and  $M = 2^A$  with  $\sqsubseteq_L = \sqsubseteq_M = \subseteq$ , then  $\beta : C \rightarrow A$  is called an **extraction function**
- $\beta$  determines **Galois connection**  $(\alpha, \gamma)$  where

$$\alpha : L \rightarrow M : I \mapsto \beta(I) (= \{\beta(c) \mid c \in I\})$$

$$\gamma : M \rightarrow L : m \mapsto \beta^{-1}(m) (= \{c \in C \mid \beta(c) \in m\})$$

## Example 13.3

- 1 Parity abstraction (cf. Example 11.2):  $\beta : \mathbb{Z} \rightarrow \{\text{even}, \text{odd}\}$  where

$$\beta(z) := \begin{cases} \text{even} & \text{if } z \text{ even} \\ \text{odd} & \text{if } z \text{ odd} \end{cases}$$

# Extraction Functions

- **Assumption:** abstraction determined by pointwise mapping of concrete elements
- If  $L = 2^C$  and  $M = 2^A$  with  $\sqsubseteq_L = \sqsubseteq_M = \subseteq$ , then  $\beta : C \rightarrow A$  is called an **extraction function**
- $\beta$  determines **Galois connection**  $(\alpha, \gamma)$  where

$$\alpha : L \rightarrow M : I \mapsto \beta(I) (= \{\beta(c) \mid c \in I\})$$

$$\gamma : M \rightarrow L : m \mapsto \beta^{-1}(m) (= \{c \in C \mid \beta(c) \in m\})$$

## Example 13.3

- ① Parity abstraction (cf. Example 11.2):  $\beta : \mathbb{Z} \rightarrow \{\text{even}, \text{odd}\}$  where

$$\beta(z) := \begin{cases} \text{even} & \text{if } z \text{ even} \\ \text{odd} & \text{if } z \text{ odd} \end{cases}$$

- ② Sign abstraction (cf. Example 11.3):  $\beta : \mathbb{Z} \rightarrow \{+, -, 0\}$  with  $\beta = \text{sgn}$



# Extraction Functions

- **Assumption:** abstraction determined by pointwise mapping of concrete elements
- If  $L = 2^C$  and  $M = 2^A$  with  $\sqsubseteq_L = \sqsubseteq_M = \subseteq$ , then  $\beta : C \rightarrow A$  is called an **extraction function**
- $\beta$  determines **Galois connection**  $(\alpha, \gamma)$  where

$$\alpha : L \rightarrow M : I \mapsto \beta(I) (= \{\beta(c) \mid c \in I\})$$

$$\gamma : M \rightarrow L : m \mapsto \beta^{-1}(m) (= \{c \in C \mid \beta(c) \in m\})$$

## Example 13.3

- ① Parity abstraction (cf. Example 11.2):  $\beta : \mathbb{Z} \rightarrow \{\text{even}, \text{odd}\}$  where

$$\beta(z) := \begin{cases} \text{even} & \text{if } z \text{ even} \\ \text{odd} & \text{if } z \text{ odd} \end{cases}$$

- ② Sign abstraction (cf. Example 11.3):  $\beta : \mathbb{Z} \rightarrow \{+, -, 0\}$  with  $\beta = \text{sgn}$
- ③ Interval abstraction (cf. Example 11.4): not definable by extraction function (as  $\text{Int}$  is not of the form  $2^A$ )

# Safe Approximation by Extraction Functions

**Reminder:** **safe approximation** condition (Definition 12.3)

$$\alpha(f(\gamma(m_1), \dots, \gamma(m_n))) \sqsubseteq_M f^\#(m_1, \dots, m_n).$$

# Safe Approximation by Extraction Functions

**Reminder:** *safe approximation* condition (Definition 12.3)

$$\alpha(f(\gamma(m_1), \dots, \gamma(m_n))) \sqsubseteq_M f^\#(m_1, \dots, m_n).$$

## Theorem 13.4

Let  $L = 2^C$  and  $M = 2^A$  with  $\sqsubseteq_L = \sqsubseteq_M = \subseteq$ ,  $\beta : C \rightarrow A$  be an extraction function, and  $f : C^n \rightarrow C$ . Then

$$f^\# : M^n \rightarrow M : (m_1, \dots, m_n) \mapsto \{\beta(f(c_1, \dots, c_n)) \mid \forall i \in \{1, \dots, n\} : c_i \in \beta^{-1}(m_i)\}$$

is a *safe approximation* of  $f$ .

# Safe Approximation by Extraction Functions

**Reminder:** *safe approximation* condition (Definition 12.3)

$$\alpha(f(\gamma(m_1), \dots, \gamma(m_n))) \sqsubseteq_M f^\#(m_1, \dots, m_n).$$

## Theorem 13.4

Let  $L = 2^C$  and  $M = 2^A$  with  $\sqsubseteq_L = \sqsubseteq_M = \subseteq$ ,  $\beta : C \rightarrow A$  be an extraction function, and  $f : C^n \rightarrow C$ . Then

$$f^\# : M^n \rightarrow M : (m_1, \dots, m_n) \mapsto \{\beta(f(c_1, \dots, c_n)) \mid \forall i \in \{1, \dots, n\} : c_i \in \beta^{-1}(m_i)\}$$

is a *safe approximation* of  $f$ .

Proof.

on the board



# Safe Approximation of Arithmetic Operations

## Example 13.5 (Sign abstraction)

For  $C = \mathbb{Z}$ ,  $A = \{+, -, 0\}$ ,  $\beta = \text{sgn}$ :

$+\#$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{+\}$	$\{+, -, 0\}$	$\{+\}$
$\{-\}$	$\{+, -, 0\}$	$\{-\}$	$\{-\}$
$\{0\}$	$\{+\}$	$\{-\}$	$\{0\}$

$*\#$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{+\}$	$\{-\}$	$\{0\}$
$\{-\}$	$\{-\}$	$\{+\}$	$\{0\}$
$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$

and  $\{+, 0\} * \# \{-\} = \{+\} * \# \{-\} \cup \{0\} * \# \{-\}$   
 $= \{-\} \cup \{0\}$   
 $= \{-, 0\}$

etc.

# Safe Approximation of Boolean Operations

## Example 13.6 (Sign abstraction)

### 1 Relational operations:

- $C = \mathbb{Z} \cup \mathbb{B}$ ,  $A = \{+, -, 0\} \cup \mathbb{B}$ ,  $\beta = \text{sgn}$

- | $=\#$   | $\{+\}$                         | $\{-\}$                         | $\{0\}$            |
|---------|---------------------------------|---------------------------------|--------------------|
| $\{+\}$ | $\{\text{true}, \text{false}\}$ | $\{\text{false}\}$              | $\{\text{false}\}$ |
| $\{-\}$ | $\{\text{false}\}$              | $\{\text{true}, \text{false}\}$ | $\{\text{false}\}$ |
| $\{0\}$ | $\{\text{false}\}$              | $\{\text{false}\}$              | $\{\text{true}\}$  |

- | $>\#$   | $\{+\}$                         | $\{-\}$                         | $\{0\}$            |
|---------|---------------------------------|---------------------------------|--------------------|
| $\{+\}$ | $\{\text{true}, \text{false}\}$ | $\{\text{true}\}$               | $\{\text{true}\}$  |
| $\{-\}$ | $\{\text{false}\}$              | $\{\text{true}, \text{false}\}$ | $\{\text{false}\}$ |
| $\{0\}$ | $\{\text{false}\}$              | $\{\text{true}\}$               | $\{\text{false}\}$ |

- $\{+, 0\} =\# \{0\} = \{+\} =\# \{0\} \cup \{0\} =\# \{0\} = \{\text{false}\} \cup \{\text{true}\} = \{\text{true}, \text{false}\}$  etc.

# Safe Approximation of Boolean Operations

## Example 13.6 (Sign abstraction)

### 1 Relational operations:

- $C = \mathbb{Z} \cup \mathbb{B}$ ,  $A = \{+, -, 0\} \cup \mathbb{B}$ ,  $\beta = \text{sgn}$

- | $=\#$   | $\{+\}$                         | $\{-\}$                         | $\{0\}$            |
|---------|---------------------------------|---------------------------------|--------------------|
| $\{+\}$ | $\{\text{true}, \text{false}\}$ | $\{\text{false}\}$              | $\{\text{false}\}$ |
| $\{-\}$ | $\{\text{false}\}$              | $\{\text{true}, \text{false}\}$ | $\{\text{false}\}$ |
| $\{0\}$ | $\{\text{false}\}$              | $\{\text{false}\}$              | $\{\text{true}\}$  |

- | $>\#$   | $\{+\}$                         | $\{-\}$                         | $\{0\}$            |
|---------|---------------------------------|---------------------------------|--------------------|
| $\{+\}$ | $\{\text{true}, \text{false}\}$ | $\{\text{true}\}$               | $\{\text{true}\}$  |
| $\{-\}$ | $\{\text{false}\}$              | $\{\text{true}, \text{false}\}$ | $\{\text{false}\}$ |
| $\{0\}$ | $\{\text{false}\}$              | $\{\text{true}\}$               | $\{\text{false}\}$ |

- $\{+, 0\} =\# \{0\} = \{+\} =\# \{0\} \cup \{0\} =\# \{0\} = \{\text{false}\} \cup \{\text{true}\} = \{\text{true}, \text{false}\}$  etc.

### 2 Boolean connectives:

- $C = A = \mathbb{B}$ ,  $\neg\# = \neg$ ,  $\wedge\# = \wedge$ , ...
- $\{\text{true}, \text{false}\} \wedge\# \{\text{true}\} = \{\text{true}\} \wedge\# \{\text{true}\} \cup \{\text{false}\} \wedge\# \{\text{true}\} = \{\text{true}\} \cup \{\text{false}\} = \{\text{true}, \text{false}\}$  etc.

**Now:** take values of variables into account

## Definition 13.7 (Abstract program state)

Let  $\beta : \mathbb{Z} \rightarrow A$  be an extraction function.

- An **abstract (program) state** is an element of the set

$$\{\rho \mid \rho : \text{Var} \rightarrow A\},$$

called the **abstract state space**.



**Now:** take values of variables into account

## Definition 13.7 (Abstract program state)

Let  $\beta : \mathbb{Z} \rightarrow A$  be an extraction function.

- An **abstract (program) state** is an element of the set

$$\{\rho \mid \rho : \text{Var} \rightarrow A\},$$

called the **abstract state space**.

- The **abstract domain** is denoted by  $Abs := 2^{\text{Var} \rightarrow A}$ .

**Now:** take values of variables into account

## Definition 13.7 (Abstract program state)

Let  $\beta : \mathbb{Z} \rightarrow A$  be an extraction function.

- An **abstract (program) state** is an element of the set

$$\{\rho \mid \rho : \text{Var} \rightarrow A\},$$

called the **abstract state space**.

- The **abstract domain** is denoted by  $Abs := 2^{\text{Var} \rightarrow A}$ .
- The **abstraction function**  $\alpha : 2^{\Sigma} \rightarrow Abs$  is given by

$$\alpha(S) := \{\beta \circ \sigma \mid \sigma \in S\}$$

for every  $S \subseteq \Sigma$ .

## Definition 13.8 (Abstract evaluation functions)

Let  $\rho : Var \rightarrow A$  be an abstract state.

- ①  $val_{\rho}^{\#} : AExp \rightarrow 2^A$  is determined by ( $f$  arithmetic operation)

$$val_{\rho}^{\#}(z) := \{\beta(z)\}$$

$$val_{\rho}^{\#}(x) := \{\rho(x)\}$$

$$val_{\rho}^{\#}(f(a_1, \dots, a_n)) := f^{\#}(val_{\rho}^{\#}(a_1), \dots, val_{\rho}^{\#}(a_n))$$

# Abstract Evaluation of Expressions

## Definition 13.8 (Abstract evaluation functions)

Let  $\rho : Var \rightarrow A$  be an abstract state.

- ①  $val_{\rho}^{\#} : AExp \rightarrow 2^A$  is determined by ( $f$  arithmetic operation)

$$val_{\rho}^{\#}(z) := \{\beta(z)\}$$

$$val_{\rho}^{\#}(x) := \{\rho(x)\}$$

$$val_{\rho}^{\#}(f(a_1, \dots, a_n)) := f^{\#}(val_{\rho}^{\#}(a_1), \dots, val_{\rho}^{\#}(a_n))$$

- ②  $val_{\rho}^{\#} : BExp \rightarrow 2^{\mathbb{B}}$  is determined by ( $g/h$  relational/Boolean op.)

$$val_{\rho}^{\#}(t) := \{t\}$$

$$val_{\rho}^{\#}(g(a_1, \dots, a_n)) := g^{\#}(val_{\rho}^{\#}(a_1), \dots, val_{\rho}^{\#}(a_n))$$

$$val_{\rho}^{\#}(h(b_1, \dots, b_n)) := h^{\#}(val_{\rho}^{\#}(b_1), \dots, val_{\rho}^{\#}(b_n))$$

# Abstract Evaluation of Expressions

## Definition 13.8 (Abstract evaluation functions)

Let  $\rho : \text{Var} \rightarrow A$  be an abstract state.

- ①  $\text{val}_\rho^\# : \text{AExp} \rightarrow 2^A$  is determined by ( $f$  arithmetic operation)

$$\text{val}_\rho^\#(z) := \{\beta(z)\}$$

$$\text{val}_\rho^\#(x) := \{\rho(x)\}$$

$$\text{val}_\rho^\#(f(a_1, \dots, a_n)) := f^\#(\text{val}_\rho^\#(a_1), \dots, \text{val}_\rho^\#(a_n))$$

- ②  $\text{val}_\rho^\# : \text{BExp} \rightarrow 2^{\mathbb{B}}$  is determined by ( $g/h$  relational/Boolean op.)

$$\text{val}_\rho^\#(t) := \{t\}$$

$$\text{val}_\rho^\#(g(a_1, \dots, a_n)) := g^\#(\text{val}_\rho^\#(a_1), \dots, \text{val}_\rho^\#(a_n))$$

$$\text{val}_\rho^\#(h(b_1, \dots, b_n)) := h^\#(\text{val}_\rho^\#(b_1), \dots, \text{val}_\rho^\#(b_n))$$

## Example 13.9 (Sign abstraction)

Let  $\rho(x) = +$  and  $\rho(y) = -$ .

①  $\text{val}_\rho^\#(2 * x + y) = \{+, -, 0\}$

②  $\text{val}_\rho^\#(\neg(x + 1 > y)) = \{\text{false}\}$

- 1 Recap: Safe Approximation of Functions and Relations
- 2 Example: Hailstone Sequences
- 3 Abstract Interpretation of WHILE Programs
- 4 Abstract Semantics of WHILE

# Abstract Semantics of WHILE I

**Reminder:** abstract domain is  $Abs := 2^{Var \rightarrow A}$

## Definition 13.1 (Abstract execution relation for statements)

If  $c \in Cmd$  and  $abs \in Abs$ , then  $\langle c, abs \rangle$  is called an **abstract configuration**. The **abstract execution relation** is defined by the following rules:

$$\text{(skip)} \frac{}{\langle \text{skip}, abs \rangle \Rightarrow \langle \downarrow, abs \rangle}$$

$$\text{(asgn)} \frac{}{\langle x := a, abs \rangle \Rightarrow \langle \downarrow, \{\rho[x \mapsto a'] \mid \rho \in abs, a' \in val_{\rho}^{\#}(a)\} \rangle}$$

$$\text{(seq1)} \frac{\langle c_1, abs \rangle \Rightarrow \langle c'_1, abs' \rangle \quad c'_1 \neq \downarrow}{\langle c_1; c_2, abs \rangle \Rightarrow \langle c'_1; c_2, abs' \rangle}$$

$$\text{(seq2)} \frac{\langle c_1, abs \rangle \Rightarrow \langle \downarrow, abs' \rangle}{\langle c_1; c_2, abs \rangle \Rightarrow \langle c_2, abs' \rangle}$$

## Definition 13.1 (Abstract execution relation for statements; cont.)

$$\text{(if1)} \frac{\exists \rho \in \text{abs} : \text{true} \in \text{val}_{\rho}^{\#}(b)}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \text{abs} \rangle \Rightarrow \langle c_1, \text{abs} \setminus \{ \rho \in \text{abs} \mid \text{val}_{\rho}^{\#}(b) = \{\text{false}\} \} \rangle}$$

$$\text{(if2)} \frac{\exists \rho \in \text{abs} : \text{false} \in \text{val}_{\rho}^{\#}(b)}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \text{abs} \rangle \Rightarrow \langle c_2, \text{abs} \setminus \{ \rho \in \text{abs} \mid \text{val}_{\rho}^{\#}(b) = \{\text{true}\} \} \rangle}$$

$$\text{(wh1)} \frac{\exists \rho \in \text{abs} : \text{true} \in \text{val}_{\rho}^{\#}(b)}{\langle \text{while } b \text{ do } c, \text{abs} \rangle \Rightarrow \langle c; \text{while } b \text{ do } c, \text{abs} \setminus \{ \rho \in \text{abs} \mid \text{val}_{\rho}^{\#}(b) = \{\text{false}\} \} \rangle}$$

$$\text{(wh2)} \frac{\exists \rho \in \text{abs} : \text{false} \in \text{val}_{\rho}^{\#}(b)}{\langle \text{while } b \text{ do } c, \text{abs} \rangle \Rightarrow \langle \downarrow, \text{abs} \setminus \{ \rho \in \text{abs} \mid \text{val}_{\rho}^{\#}(b) = \{\text{true}\} \} \rangle}$$



## Definition 13.2 (Abstract transition function)

The **abstract transition function** is defined by the family of mappings

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

given by

$$\text{next}_{c,c'}^{\#}(abs) := \bigcup \{abs' \in Abs \mid \langle c, abs \rangle \Rightarrow \langle c', abs' \rangle\}$$

# Abstract Semantics of WHILE III

## Definition 13.2 (Abstract transition function)

The **abstract transition function** is defined by the family of mappings

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

given by

$$\text{next}_{c,c'}^{\#}(abs) := \bigcup \{abs' \in \text{Abs} \mid \langle c, abs \rangle \Rightarrow \langle c', abs' \rangle\}$$

## Example 13.3 (Hailstone Sequences; cf. Example 13.1)

```
[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;
```

Execution relation with parity abstraction: see following slide (courtesy B. König)

# Abstrakte Interpretation von HAILSTONE

