Foundations of Abstract Interpretation, Lecture 3

Taken from Lecture II of David Schmidt.
Galois Insertion

A Galois connection \( \langle C, \alpha, \gamma, A \rangle \) is called a Galois insertion, if \( \alpha \circ \gamma = \lambda a.a \). Every Galois connection can be lifted to a Galois insertion, by identifying every two abstract elements which have the same concretization.

Assume that \( \langle \mathcal{P}(C), \alpha, \gamma, A \rangle \) is a Galois connection which is not an insertion. In this case, there must exists two distinct abstract elements \( a_1 \neq a_2 \), such that \( a_2 = \alpha \circ \gamma(a_1) \). Composing \( \gamma \) to both sides of the equality, we obtain \( \gamma(a_2) = \gamma \circ \alpha \circ \gamma(a_1) = \gamma(a_1) \). Thus, a non-insertion connection is possible only if we have two distinct abstract elements with the same \( \gamma \)-mapping.

Let \( \langle C, \alpha, \gamma, A \rangle \) be a non-insertion connection. Define \( A_\sim \) to be a new abstract domain whose elements are equivalence sets over \( A \) where \( 2 \) \( A \)-elements are equivalent if they have the same \( \gamma \)-mapping. For each \( a_\sim \in A_\sim \), we define \( \gamma_\sim(a_\sim) = \gamma(a) \), where \( a \) is any member of \( a_\sim \).

Ordering over \( A_\sim \) is defined by \( a_\sim \sqsubseteq b_\sim \iff \gamma_\sim(a_\sim) \subseteq \gamma_\sim(b_\sim) \). For every \( S \subseteq C \), the adjunct mapping can be defined by

\[
\alpha_\sim(S) = \gamma_\sim^{-1}(\bigcap\{T \mid S \subseteq T \subseteq C\}) = \sqcap\{a_\sim \in A_\sim \mid S \subseteq \gamma_\sim(a_\sim)\}
\]

It is not difficult to show that \( \langle \mathcal{P}(C), \alpha_\sim, \gamma_\sim, A_\sim \rangle \) is a Galois insertion.
Subset Structured Abstract Domains

Earlier, we restricted our attention to the case that the concrete domain has the structure $\mathcal{P}(\mathcal{C})$. For such a case, it is possible to specify a Galois connection by a relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{A}$, provided it satisfies:

$$c \mathcal{R} a \land a \subseteq b \rightarrow c \mathcal{R} b \quad \text{and} \quad c \mathcal{R} \cap \{a \mid c \mathcal{R} a\}$$

An additional useful restriction is the case that the abstract domain also has the structure $\mathcal{P}(\mathcal{A})$, where $\mathcal{A}$ is an unordered domain of individuals. In this case, it is sufficient to specify an abstraction function $\beta : \mathcal{C} \mapsto \mathcal{A}$ in order to obtain a Galois connection. Such a function induces a relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{P}(\mathcal{A})$ given by $\beta(c, \mathcal{A}) : \beta(c) \in \mathcal{A}$.

The above two closure requirements are translated into the following valid properties:

$$\beta(c) \in \mathcal{A} \land \mathcal{A} \subseteq \mathcal{B} \rightarrow \beta(c) \in \mathcal{B} \quad \text{and} \quad \beta(c) \in \bigcap\{\mathcal{A} \mid \beta(c) \in \mathcal{A}\}$$

It follows that the specification of a structure $\langle \mathcal{C}, \beta, \mathcal{A} \rangle$ gives rise to the Galois connection $\langle \mathcal{P}(\mathcal{C}), \alpha, \gamma, \mathcal{P}(\mathcal{A}) \rangle$, where

- $\alpha(S) : \{\beta(c) \mid c \in S\}$, for every $S \subseteq \mathcal{C}$.
- $\gamma(A) : \{c \mid \beta(c) \in A\}$, for every $A \subseteq \mathcal{A}$. 
Concrete and abstract operations

Now that we know how to model \( c \in C \) by \( \alpha(c) \in \mathcal{A} \), we must model concrete computation steps, \( f : C \to C \), by abstract computation steps, \( f^\# : \mathcal{A} \to \mathcal{A} \).

**Example:** We have concrete domain, \( \text{Nat} \), and concrete operation, \( \text{succ} : \text{Nat} \to \text{Nat} \), defined as \( \text{succ}(n) = n + 1 \).

We have abstract domain, \( \text{Parity} \), and abstract operation, \( \text{succ}^\# : \text{Parity} \to \text{Parity} \), defined as

\[
\begin{align*}
\text{succ}^\#(\text{even}) &= \text{odd}, & \text{succ}^\#(\text{odd}) &= \text{even} \\
\text{succ}^\#(\text{any}) &= \text{any}, & \text{succ}^\#(\text{none}) &= \text{none}
\end{align*}
\]

\( \text{succ}^\# \) must be consistent (sound) with respect to \( \text{succ} \):

\[
\text{if } n \mathcal{R}_{\text{Nat}} a, \text{ then } \text{succ}(n) \mathcal{R}_{\text{Nat}} \text{succ}^\#(a)
\]

where \( \mathcal{R} \subseteq \text{Nat} \times \text{Parity} \) relates numbers to their parities (e.g., \( 2 \mathcal{R}_{\text{Nat}} \text{even}, 5 \mathcal{R}_{\text{Nat}} \text{odd}, \) etc.).
We want *soundness*: \( n \mathcal{R}_{\text{Nat}} a \) implies \( \text{succ}(n) \mathcal{R}_{\text{Nat}} \text{succ}^\#(a) \), for all \( n \in \text{Nat} \) and \( a \in \text{Parity} \).

Since we have the Galois connection, \( \wp(\text{Nat}) \langle \alpha, \gamma \rangle \text{Parity} \), we know that \( \gamma(a) = \{ n \mid n \mathcal{R}_{\text{Nat}} a \} \).

So, soundness is stated equivalently as

\[
\text{for all } a \in A, \text{ for all } n \in \gamma(a), \text{succ}(n) \in \gamma(\text{succ}^\#(a))
\]

and this is equivalent to saying,

\[
\text{for all } a \in A, \text{succ}^* (\gamma(a)) \subseteq_{\text{Nat}} \gamma(\text{succ}^\#(a))
\]

that is,

\[
\text{for all } a \in A, (\text{succ}^* \circ \gamma)(a) \subseteq_{\text{Nat}} (\gamma \circ \text{succ}^\#)(a)
\]

where \( \text{succ}^* : \wp(\text{Nat}) \to \wp(\text{Nat}) \) is \( \text{succ}^*(S) = \{ \text{succ}(n) \mid n \in S \} \).

This is interesting, because it states a commutative, “semi-homomorphism” property....
Definition: For Galois connection, $\mathcal{C}(\alpha, \gamma)\mathcal{A}$, and functions $f : C \rightarrow C$, $f^\# : A \rightarrow A$, $f^\#$ is a **sound approximation** of $f$ iff

$$(\alpha \circ f)(c) \sqsubseteq A (f^\# \circ \alpha)(c), \text{ for all } c \in C$$

iff

$$(f \circ \gamma)(a) \sqsubseteq C (\gamma \circ f^\#)(a), \text{ for all } a \in A$$

This slightly abstract presentation exposes that $\alpha$ is a “semi-homomorphism” with respect to $f$ and $f^\#$:

$$
\begin{array}{c}
c \\
\downarrow f \\
f(c) \\
\end{array} \xrightarrow{\alpha} \alpha(c) \xrightarrow{f^\#} \alpha(f(c)) \sqsubseteq f^\#(\alpha(c))
$$
Example 1: \( n \mathcal{R}_{\text{Nat}} \alpha \text{ implies } \text{succ}(n) \mathcal{R}_{\text{Nat}} \text{succ}^\#(a) \)

Galois connection: \( \wp(\text{Nat}) \langle \alpha, \gamma \rangle \text{Parity} \)

\[
\text{succ}^*: \wp(\text{Nat}) \rightarrow \wp(\text{Nat})
\]

\[
\text{succ}^*(S) = \{ \text{succ}(n) \mid n \in S \}
\]

where \( \text{succ}(n) = n + 1 \)

\[
\text{succ}^\#: \text{Parity} \rightarrow \text{Parity}
\]

\[
\text{succ}^\#(\text{even}) = \text{odd}, \quad \text{succ}^\#(\text{odd}) = \text{even}
\]

\[
\text{succ}^\#(\text{any}) = \text{any}, \quad \text{succ}^\#(\text{none}) = \text{none}
\]

We have that \( \alpha \circ \text{succ}^* = \text{succ}^\# \circ \alpha: \)

\[
\begin{align*}
\{2,3\} & \xrightarrow{\alpha} \text{any} \\
\text{succ}^* & \xrightarrow{\alpha} \{3,4\} \xrightarrow{\text{any}} \text{any} = \text{any} \\
\text{succ}^\# & \xrightarrow{\text{any}} \text{any}
\end{align*}
\]
**Example 2:** \( n \mathcal{R}_{\text{Nat}} \alpha \) implies \( \text{div}2(n) \mathcal{R}_{\text{Nat}} \text{div}2\#(\alpha) \)

Galois connection: \( \wp(\text{Nat})\langle \alpha, \gamma \rangle \text{Parity} \)

\[
\text{div}2^*: \wp(\text{Nat}) \to \wp(\text{Nat})
\]

\[
\text{div}2^*(S) = \{ \text{div}2(n) \mid n \in S \}
\]

where \( \text{div}2(2n + 1) = \text{div}2(2n) = n \)

\[
\text{div}2\#: \text{Parity} \to \text{Parity}
\]

\[
\text{div}2\#(\text{even}) = \text{div}2\#(\text{odd}) = \text{any}
\]

\[
\text{div}2\#(\text{any}) = \text{any}, \quad \text{div}2\#(\text{none}) = \text{none}
\]

We have that \( \alpha \circ \text{div}2^* \sqsubseteq_{\text{Parity}} \text{div}2\# \circ \alpha \):

\[
\begin{align*}
\{6\} & \xrightarrow{\alpha} \text{even} \\
\text{div}2^* & \downarrow \\
\{3\} & \xrightarrow{\alpha} \text{odd} \sqsubseteq \text{any} \\
\text{div}2\# & \downarrow
\end{align*}
\]
Lifting Functions For Subset Structured Domains

Consider a subset-structured abstraction \( \langle C, \beta, A \rangle \). With each operation \( f : C \mapsto C \), we associate a corresponding abstract operation \( f^\# : A \mapsto \mathcal{P}(A) \). Both \( f \) and \( f^\# \) are extended point-wise to set arguments.

The abstract operation \( f^\# \) is a sound approximation of \( f \) iff

\[
\beta \circ f(c) \in f^\# \circ \beta(c), \text{ for all } c \in C
\]

iff

\[
f \circ \beta^{-1}(a) \subseteq \beta^{-1} \circ f^\#(a), \text{ for all } a \in A
\]
Synthesizing $f\#$ from $f$

The previous slides show how $\alpha$ acts as a “semi-homomorphism” between $f$ and $f\#$.

Given the Galois connection, $\mathcal{C}(\alpha, \gamma)\mathcal{A}$, and operation, $f : \mathcal{C} \rightarrow \mathcal{C}$, the most precise $f\#_{\text{best}} : \mathcal{A} \rightarrow \mathcal{A}$ that is sound with respect to $f$ is

$$f\#_{\text{best}} = \alpha \circ f \circ \gamma$$

**Proposition:** $f\#$ is sound with respect to $f$ iff $f\#_{\text{best}} \sqsubseteq_{\mathcal{A} \rightarrow \mathcal{A}} f\#$.

(Note: $f \sqsubseteq_{\mathcal{A} \rightarrow \mathcal{A}} g$ iff for all $a \in \mathcal{A}$, $f(a) \sqsubseteq_{\mathcal{A}} g(a)$.)

Of course, $f\#_{\text{best}}$ has a *mathematical* definition — not an algorithmic one — $f\#_{\text{best}}$ might not be finitely computable!

$$\text{succ}_{\text{best}}(\text{even}) = \alpha \circ \text{succ}^\ast(\gamma \text{ even})$$

**Parity example continued:**

$$= \alpha(\text{succ}^\ast\{2n \mid n \geq 0\})$$

$$= \alpha\{2n + 1 \mid n \geq 0\} = \text{odd}$$
One more example:

Given \(\varphi(\text{Nat}) \langle \alpha, \gamma \rangle \text{Parity} \) and \(\text{div}2 : \text{Nat} \rightarrow \text{Nat} \), we have

\[
\text{div}2^* : \varphi(\text{Nat}) \rightarrow \varphi(\text{Nat})
\]

\[
\text{div}2^*(S) = \text{div}2[S] = \{\text{div}2(n) \mid n \in S\}
\]

Hence, \(\text{div}2^\#_{\text{best}} = \alpha \circ \text{div}2^* \circ \gamma\). The operation loses precision:

\[
\alpha(\text{div}2^*[4]) = \alpha[2] = \text{even}, \text{ but}
\]

\[
\text{div}2^\#_{\text{best}}(\text{even}) = \alpha(\text{div}2^*(\gamma(\text{even})))
\]

\[
= \alpha(\text{div}2^*[0, 2, 4, \ldots])
\]

\[
= \alpha[1, 2, 3, \ldots] = \text{any}
\]

Nonetheless, this is the best we can do, given the crude structure of the abstract domain, \textbf{Parity}.
The Best Sound Approximation

Claim 5. The function $f_{\text{best}}^\# = \alpha \circ f \circ \gamma$ is a sound approximation of $f$ and is the smallest sound approximation.

Proof: To show that $f_{\text{best}}^\#$ is sound, we have to show $f \circ \gamma \subseteq \gamma \circ f_{\text{best}}^\#$. Observe that

$$\gamma \circ f_{\text{best}}^\# = \gamma \circ \alpha \circ f \circ \gamma = f \circ \gamma$$

since we are considering insertions in which $\gamma \circ \alpha = \lambda a.a$.

Let $f^\#$ be any (other) sound abstraction. Being sound, it satisfies $f \circ \gamma \subseteq \gamma \circ f^\#$. Composing both sides with $\alpha$ on the left and canceling $\alpha \circ g$, we obtain $f_{\text{best}}^\# = \alpha \circ f \circ \gamma \subseteq f^\#$.

For the case of subset-structured abstractions, $f_{\text{best}}^\#$ is given by $\beta \circ f \circ \beta^{-1}$.
Completeness

Given $C\langle \alpha, \gamma \rangle A$, we state soundness of $f^#$ with respect to $f$ as
$
\alpha \circ f \sqsubseteq_{A \rightarrow A} f^# \circ \alpha \iff f \circ \gamma \sqsubseteq_{C \rightarrow C} \gamma \circ f^#
$

**Definition:** $f^#$ is *forwards* $(\gamma)$ *complete* with respect to $f$ iff
$f \circ \gamma =_{C \rightarrow C} \gamma \circ f^#
$

**Definition:** $f^#$ is *backwards* $(\alpha)$ *complete* with respect to $f$ iff
$\alpha \circ f =_{A \rightarrow A} f^# \circ \alpha
$

The two completeness notions are not equivalent!

For an $f^#$ to be (forwards or backwards) complete, it must equal

$f^\#_{\text{best}} = \alpha \circ f \circ \gamma$. Indeed, the structure of the Galois connection and $f : C \rightarrow C$ determines whether $f^\#_{\text{best}}$ is complete.
Forwards ($\gamma$) completeness: $f^{\#}_{\text{best}}$ is forwards-complete iff $f$ maps image points of $\gamma$ to image points of $\gamma$ — $f(\gamma[A]) \subseteq \gamma[A]$.

Backwards ($\alpha$) completeness: $f^{\#}_{\text{best}}$ is backwards-complete iff $f$ maps all points in the same $\alpha$-equivalence class to points in the same $\alpha$-equivalence class — $\alpha(c) = \alpha(c')$ implies $\alpha(f(c)) = \alpha(f(c'))$. 
### Illustrating Forwards and Backwards Completeness

**Forwards ($\gamma$) Completeness:** $f^\#_{best}$ is forwards-complete iff $f$ maps image points of $\gamma$ to image points.

- $\text{succ}^\#_{best}$ is not forwards complete, because $\text{succ}(\text{odd}) = \{2, 4, \ldots\} \subset \text{even}$.
- $\text{div2}^\#_{best}$ is forwards complete, because $\text{div2}(\text{odd}) = \text{div2}(\text{even}) = \{0, 1, 2, \ldots\} = \text{any}$.

**Backwards ($\alpha$) Completeness:** $f^\#_{best}$ is backwards-complete iff $f$ maps all points in the same $\alpha$-equivalence class to points in the same $\alpha$-equivalence class iff $f^\#_{best}$ maps individuals in $\mathcal{A}$ to individuals.

- $\text{succ}^\#_{best}$ is backwards complete, because $\text{succ}^\#_{best}(\text{odd}) = \text{even}$ and $\text{succ}^\#_{best}(\text{even}) = \text{odd}$.
- $\text{div2}^\#_{best}$ is not backwards complete, because

  $$\alpha(\text{div2}(4)) = \alpha(2) = \text{even} \neq \alpha(\text{div2}(6)) = \alpha(3) = \text{odd},$$

  even though $a(4) = \alpha(6) = \text{even}$.
**Transfer functions** generate computation steps

Each program transition from program point $p_i$ to $p_j$ has an associated *transfer function*, $f_{ij} : C \rightarrow C$ (or $f^\#_{ij} : A \rightarrow A$), which describes the associated computation.

This defines a computation step of the form, $p_i, s \rightarrow p_j, f_{ij}(s)$.

**Example:** Assignment $p_0 : x = x + 1$; $p_1 : \cdots$ has the transfer function, $f_{01}(\ldots x : n\ldots) = (\ldots x : n + 1\ldots)$. For example, $p_0, \langle x : 3 \rangle \rightarrow p_1, f_{01}(\langle x : 3 \rangle) = p_1, \langle x : 4 \rangle$.

For modelling multiple transitions in conditional/nondeterministic choice, we attach a transfer function to each possible transition.

\[
\begin{align*}
p_0 : \text{cases} \\
x \leq y : & \quad p_1 : y = y - x; \\
y \leq x : & \quad p_2 : x = x - y; \\
\text{end}
\end{align*}
\]
For
\[ p_0 : \text{cases} \]
\[ x \leq y : \ p_1 : y = y - x; \]
\[ y \leq x : \ p_2 : x = x - y; \]
end

we have these functions:
\[ f_{01}(s) = \begin{cases} s & \text{if } s.x \leq s.y \\ \bot & \text{otherwise} \end{cases} \]
\[ f_{02}(s) = \begin{cases} s & \text{if } s.y \leq s.x \\ \bot & \text{otherwise} \end{cases} \]

For example, \( p_0, \langle x : 5, y : 3 \rangle \rightarrow p_1, \bot \), because \( x \not\leq y \), but
\( p_0, \langle x : 5, y : 3 \rangle \rightarrow p_2, \langle x : 5, y : 3 \rangle \), because \( y \leq x \). The transfer functions “filter” the data that arrives at a program point.

We ignore computation steps, \( p, s \rightarrow p', \bot \), that produce “no data” (\( \bot \)).

An execution trace is a (possibly infinite) sequence,
\( p_0, s_0 \rightarrow p_1, s_1 \rightarrow \cdots \rightarrow p_j, s_j \rightarrow \cdots \), such that, for all \( i \geq 0 \):
\[ \Diamond \ p_i, s_i \rightarrow p_{\text{succ}(i)}, f_{i, \text{succ}(i)}(s_i) \]
\[ \Diamond \ \text{no } s_i \text{ equals } \bot. \]
Using the f#'s to build sound, abstract trace trees

\[ p_0 : \text{while } (x \neq 1) \{ \]
\[ p_1 : \text{if } \text{Even}(x) \]
\[ p_2 : \text{then } x = x \text{ div} 2; \]
\[ p_3 : \text{else } x = 3\times x + 1; \]
\[ \} \]
\[ p_4 : \text{exit} \]

**Note:** \( p_i, v \) abbreviates \( p_i, \langle x : v \rangle \)

Abstract overapproximating trace:

```
  \begin{array}{c}
    p_0, \text{even} \\
    p_0, \text{any} \\
    p_4, \text{odd} \\
    p_4, \text{any} \\
    p_1, \text{even} \\
    p_2, \text{even} \\
    \end{array}
```

Two concrete traces:

```
  \begin{array}{c}
    p_0, 4 \\
    p_1, 4 \\
    p_2, 4 \\
    p_0, 2 \\
    p_1, 2 \\
    p_2, 2 \\
    p_0, 1 \\
    p_4, 1 \\
  \end{array}
```

Each concrete transition is generated by an \( f_{ij} \); each abstract transition is generated by the corresponding \( f_i# \).
Each concrete transition, $p_i, s \rightarrow p_j, f_{ij}(s)$, is reproduced by a corresponding abstract transition, $p_i, a \rightarrow p_j, f^\#_{ij}(a)$, where $s \in \gamma(a)$.

**For example**, $p_2 : x = x \div 2$ is interpreted *concretely* by $f_{20}(2n) = n = f_{20}(2n + 1)$ and is interpreted *abstractly* by $f^\#_{20}(\text{even}) = \text{any} = f^\#_{20}(\text{odd}) = f^\#_{20}(\text{any})$.

The traces embedded in the abstract trace tree “cover” (*simulate*) the concrete traces, e.g., this concrete trace,

$p_0, 4 \rightarrow p_1, 4 \rightarrow p_2, 4 \rightarrow p_0, 2 \rightarrow p_1, 2 \rightarrow p_2, 2 \rightarrow p_0, 1 \rightarrow p_4, 1$

is simulated by this abstract trace, which is extracted from the abstract computation tree:

$p_0, \text{even} \rightarrow p_1, \text{even} \rightarrow p_2, \text{even} \rightarrow p_0, \text{any} \rightarrow p_1, \text{any} \rightarrow p_2, \text{even} \rightarrow p_0, \text{any} \rightarrow p_4, \text{odd}$

and indeed, *all* concrete traces starting with $p_0, 2n$, $n \geq 0$, are simulated by the abstract tree in this manner.
Proof of soundness of trace construction

For $S \in C$ and $a \in A$, say that $S \mathcal{R} a$ iff $S \subseteq_C \gamma(a)$ iff $\alpha(S) \subseteq_A a$.

**Lemma:** $\alpha \circ f \subseteq_A \alpha \circ f^\# \circ \alpha \iff f \circ \gamma \subseteq_C \gamma \circ f^\#$ iff $S \mathcal{R} a$ implies $f(S) \mathcal{R} f^\#(a)$.

**Theorem:** For every concrete trace, $(p_i, s_i)_{i \geq 0}$, there exists an abstract trace, $(p_i, a_i)_{i \geq 0}$, such that for all $i \geq 0$, $\{s_i\} \mathcal{R} a_i$.

**Proof:** We use the Lemma and induction to assemble this diagram:

$p_0, s_0 \rightarrow p_1, f_0(s_0) = p_1, s_1 \rightarrow p_2, f_1(s_1) = p_2, s_2 \rightarrow \cdots \rightarrow p_i, s_i \rightarrow \cdots$

$\mathcal{R}$  $\mathcal{R}$  $\mathcal{R}$  $\mathcal{R}$  $\mathcal{R}$

$p_0, a_0 \rightarrow p_1, f_0^\#(a_0) = p_1, a_1 \rightarrow p_2, f_1^\#(a_1) = p_2, a_2 \rightarrow \cdots \rightarrow p_i, a_i \rightarrow \cdots$

(Note: each $s_i$ in the diagram is more precisely stated as $\{s_i\}$, because $C = \varnothing(Store)$. Due to imprecision of the $f^\#$s, the abstract trace tree may possess many traces that begin with $p_0, a_0$, but there is always one trace in the tree that simulates the concrete trace.)
When all the operations, \( f_{ij}^{\#} \), are complete with respect to the \( f_{ij} \)s, the previous result is strengthened:

Say that \( S \mathcal{R} a \iff \alpha(S) = a \). (Similarly, say that \( S \mathcal{R} a \iff S = \gamma(a) \).)

In both cases, the lemma holds:

**Lemma:** \( \alpha \circ f \models_{\lambda \rightarrow \lambda} f^{\#} \circ \alpha \iff S \mathcal{R} a \impliedby f(S) \mathcal{R} f^{\#}(a) \).
(Similarly, \( f \circ \gamma \models_{\kappa \rightarrow \kappa} \gamma \circ f^{\#} \iff S \mathcal{R} a \impliedby f(S) \mathcal{R} f^{\#}(a) \).)

**Theorem** (\( \alpha \)-completeness): When \( S \mathcal{R} a \iff \alpha(S) = a \), then for every concrete trace, \( (p_i, s_i)_{i \geq 0} \), there exists an abstract trace, \( (p_i, a_i)_{i \geq 0} \), such that for all \( i \geq 0 \), \( \{s_i\} \mathcal{R} a_i \).

**Theorem** (\( \gamma \)-completeness): When \( S \mathcal{R} a \iff \gamma(a) = S \), \( S \subseteq \text{Store} \), then for every trace on *sets of stores*, \( (p_i, S_i)_{i \geq 0} \), there exists an abstract trace, \( (p_i, a_i)_{i \geq 0} \), such that for all \( i \geq 0 \), \( S_i \mathcal{R} a_i \).
References


