Data Abstraction

The Common Wisdom:

To verify a reactive system $S$,

- If it is finite state, model check it.

- Otherwise, prove it by temporal deduction, using a temporal deductive system such as [MP] or TLA, supported by theorem provers, such as STeP, TLP, or PVS.

Often, both approaches to verification can be simplified by using abstraction.
AAV: Abstraction Aided Verification

An Obvious idea:

- **Abstract** system $S$ into $S_A$ – a simpler system, but admitting more behaviors.

- **Verify** property for the abstracted system $S_A$.

- **Conclude** that property holds for the concrete system.

Approach is particularly **impressive** when abstracting an **infinite-state** system into a **finite-state** one.
Can Abstraction Replace Deduction?

An intriguing question is whether abstraction can completely replace the need for Temporal Deduction.

That is, is it the case that, for every (possibly infinite-state) system $D$ and a property $\psi$ valid for $D$, we can find an abstraction $\alpha$ such that $D^\alpha$ is finite-state, $\psi^\alpha$ is propositional, and $D^\alpha \models \psi^\alpha$?

This will relegate temporal reasoning to the regime of automatic model checking techniques.

Possible Advantages:

- First-order temporal reasoning is more difficult to master than first-order state reasoning.

- People (engineers) find it easier to program, than to write logical formulas. An abstraction is easier to develop (program) than an invariant assertion.
Finitary Abstraction

Based on the notion of abstract interpretation [CC77].

Let $\Sigma$ denote the set of states of an FDS $\mathcal{D}$ – the concrete states. Let $\alpha : \Sigma \mapsto \Sigma_A$ be a mapping of concrete into abstract states. $\alpha$ is finitary if $\Sigma_A$ is finite.

We formulate the strategy of Verification by finitary Abstraction:

- Define a finitary abstraction mapping $\alpha$ to abstract the (possibly infinite) concrete FDS $\mathcal{D}$ into a finite-state abstract FDS $\mathcal{D}^\alpha$.

- Abstract the temporal property $\psi$ into a finitary abstract property $\psi^\alpha$.

- Model Check $\mathcal{D}^\alpha \models \psi^\alpha$.

- Conclude $\mathcal{D} \models \psi$.

The question is how to define the abstractions $\mathcal{D}^\alpha$ and $\psi^\alpha$ such that $\mathcal{D}^\alpha \models \psi^\alpha$ implies $\mathcal{D} \models \psi$?

That is, how to ensure that the abstraction is sound.
Example: Program ANY-Y

Consider the program

\[ x, y : \text{integer initially } x = y = 0 \]

\[
P_1 :: \begin{cases} 
\ell_0 : \text{while } x = 0 \text{ do} \\
\ell_1 : y := y + 1 \\
\ell_2 : 
\end{cases} \quad \parallel \quad P_2 :: \begin{cases} 
m_0 : x := 1 \\
m_1 : 
\end{cases}
\]

Assume we wish to verify the property \(\Box (y \geq 0)\) for system ANY-Y.

We introduce two abstract variables:

\[ X : \text{boolean}, \quad Y : \{-1, 0, +1\} \]

The abstraction mapping \(\alpha\) is specified by the following list of defining expressions:

\[ \alpha : [X = (x \neq 0), \quad Y = \text{sign}(y)] \]

where \(\text{sign}(y)\) is defined to be \(-1, 0,\) or \(1\), according to whether \(y\) is negative, zero, or positive, respectively.
The Abstracted Version

With the mapping $\alpha$, we can obtain the abstract version of $\text{ANY-Y}$, called $\text{ANY-Y}^\alpha$:

$$\begin{align*}
X & : \text{boolean} \quad \text{where } X = 0 \\
Y & : \{-1, 0, 1\} \quad \text{where } Y = 0
\end{align*}$$

$$P_1 :: \begin{cases}
\ell_0 : \text{while } X = 0 \text{ do} \\
\ell_1 : Y := \begin{cases}
\text{if } & Y = -1 \\
\text{then } & \{-1, 0\} \\
\text{else } & 1
\end{cases} \\
\ell_2 : &
\end{cases} \parallel P_2 :: \begin{cases}
 m_0 : X := 1 \\
m_1 : &
\end{cases}$$

The original invariance property $\psi : \square (y \geq 0)$, is abstracted into:

$$\psi^\alpha : \square (Y \in \{0, 1\})$$

which can be model-checked over $\text{ANY-Y}^\alpha$. 
Abstraction of Assertions

Assume that the abstraction mapping is given as \( V_A = \mathcal{E}^\alpha(V) \). How to lift \( \alpha \) from states to assertions?

There are two (dual) ways to abstract an assertion \( p \):

\[
\alpha^+(p): \quad \exists V \ (V_A = \mathcal{E}^\alpha(V) \land p(V))
\]
\[
\alpha^-(p): \quad \text{map}(V_A) \land \forall V \ (V_A = \mathcal{E}^\alpha(V) \rightarrow p(V)) \quad \text{where}
\]
\[
\text{map}(V_A): \quad \exists V : \ V_A = \mathcal{E}^\alpha(V)
\]

Observe:

- An abstract state \( S \in \Sigma_A \) satisfies \( \alpha^+(p) \) iff some concrete state \( s \in \alpha^{-1}(S) \) satisfies \( p \).
- An abstract state \( S \in \Sigma_A \) satisfies \( \alpha^-(p) \) iff \( \alpha^{-1}(S) \neq \emptyset \) and all concrete states \( s \in \alpha^{-1}(S) \) satisfy \( p \).

Equivalently:

\[
\alpha^{-1}(\|\alpha^-(p)\|) \subseteq \|p\| \subseteq \alpha^{-1}(\|\alpha^+(p)\|)
\]

contracting \hspace{5cm} expanding
The Two Abstraction Transformers
The Existential (expanding) Abstraction

An abstract state $S$ belongs to $\alpha^+(p)$ if some concrete state $\alpha$-mapped into $S$ satisfies $p$.

For example,

$$\alpha^+(0 \leq y \leq 5) = \exists y : (Y = \text{sign}(y) \land 0 \leq y \leq 5)$$

$$\sim Y \in \{0, 1\}$$

$$\alpha^{-1}(\|Y \in \{0, 1\}\|) = \{y \mid y \geq 0\} \supseteq \{y \mid 0 \leq y \leq 5\}$$
The Universal (contracting) Abstraction

Abstract state $S$ belongs to $\alpha^-(p)$ if all concrete states $\alpha$-mapped into $S$ satisfy $p$. For example,

\[
\begin{align*}
\alpha^-(0 \leq y \leq 5) &= \exists y : (Y = \text{sign}(y)) \land \forall y : (Y = \text{sign}(y) \rightarrow 0 \leq y \leq 5) \\
&\sim Y = 0 \\
\alpha^{-1}(\|Y = 0\|) &= \{0\} \subseteq \{y \mid 0 \leq y \leq 5\}
\end{align*}
\]
Which Should we use?

For abstracting the transition relation, we use $\alpha^+$, while for abstracting the property, we use $\alpha^-$.

An informal argument is that, in order to increase

$$\|D\| \cap \|\neg \psi\| = \|D\| \cap \|\psi\|,$$

we should expand $\|D\|$ while contracting $\|\psi\|$. 
**Additional Evidence**

Consider two assertions which are candidates for invariants of program ANY-Y, and their possible abstractions:

<table>
<thead>
<tr>
<th></th>
<th>$p$</th>
<th>$\alpha^+(p)$</th>
<th>$\alpha^-(p)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>$y \geq 0$</td>
<td>$Y \in {0,1}$</td>
<td>$Y \in {0,1}$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>$0 \leq y \leq 5$</td>
<td>$Y \in {0,1}$</td>
<td>$Y = 0$</td>
</tr>
</tbody>
</table>

It is not difficult to see that program ANY-$Y^\alpha$ has essentially only two $Y$-observations which are given by

$\langle Y : 0 \rangle$, $\langle Y : 0 \rangle$, $\langle Y : 0 \rangle$, ... and $\langle Y : 0 \rangle$, ..., $\langle Y : 0 \rangle$, $\langle Y : 1 \rangle$, $\langle Y : 1 \rangle$, ...

The following table indicates which of the assertions $p_i$, $\alpha^+(p_i)$, $\alpha^-(p_i)$, is a valid invariant over its respective system.

<table>
<thead>
<tr>
<th></th>
<th>$p$</th>
<th>$\alpha^+(p)$</th>
<th>$\alpha^-(p)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>valid</td>
<td>valid</td>
<td>valid</td>
</tr>
<tr>
<td>$p_2$</td>
<td>invalid</td>
<td>valid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

In a sound abstraction method $\mathcal{D}^\alpha \models p^\alpha$ should imply $\mathcal{D} \models p$. Consequently, only $\alpha^-(p)$ is a feasible candidate for the abstraction to be applied to the property $p$. 
Sound Temporal Abstraction

For a temporal formula $\psi$, let $\psi^\alpha$ be obtained from $\psi$ by replacing

- Every maximal $p$, a sub-assertion of $\psi$ of positive polarity (enclosed within an even number of negations), by $\alpha^{-}(p)$, and

- Every maximal $p$, a sub-assertion of $\psi$ of negative polarity (enclosed within an odd number of negations), by $\alpha^{+}(p)$.

Then

**Claim 12. [Soundness of an LTL Abstraction]**

*For a system $\mathcal{D}$ and temporal property $\psi$,*

\[ \models \psi^\alpha \quad \text{implies} \quad \models \psi. \]

This claim is based on the observation (provable by induction on the structure of $\psi$) that, for every state sequence $\sigma : s_0, s_1, \ldots$, and every position $j \geq 0$,

\[ (\alpha(\sigma), j) \models \psi^\alpha \quad \text{implies} \quad (\sigma, j) \models \psi. \]

Let us provide more details about the proof of Claim 12. With no loss of generality, assume that $\psi$ uses the boolean operators $\lor, \land, \neg$ and the temporal operators $\bigcirc, \square, \Diamond, \mathcal{U}, \mathcal{V}$, where $p \mathcal{V} q = \neg((\neg p) \mathcal{U} (\neg q))$. 
Proof of Soundness Continued

Consider first the case that $\psi$ is a formula in a positive form, i.e., that negations can only appear within state sub-formulas. Let $p_1, \ldots, p_n$ be all the maximal sub-assertions of $\psi$ which, by definition, have each a positive polarity. Writing $\psi = \psi(p_1, \ldots, p_m)$, we have that $\psi^\alpha = \psi(\alpha^-(p_1), \ldots, \alpha^-(p_m))$. We proceed to show by induction on the size of $\psi$ that $(\alpha(\sigma), j) \models \psi^\alpha$ implies $(\sigma, j) \models \psi$. If $\sigma = s_0, s_1, \ldots$, denote $\alpha(\sigma) = S_0, S_1, \ldots$, where $S_j = \alpha(s_j)$, for every $j = 0, 1, \ldots$

- For the case that $\psi$ is an assertion, i.e., $\psi = p_i$, then $(\alpha(\sigma), j) \models \psi$ implies $S_j \models \alpha^-(p_i)$. Recall that this is the case if $S_j$ has at least one $\alpha$-source and $s \models p_i$ for every $s$ such that $\alpha(s) = S_j$. Since $\alpha(s_j) = S_j$, we conclude that $s_j \models p_i$ and, therefore $(\sigma, j) \models p_i$, leading to $(\sigma, j) \models \psi$

- For the case that $\psi = p \land q$, then since $(p \land q)^\alpha = p^\alpha \land q^\alpha$, $(\alpha(\sigma), j) \models \psi^\alpha$ implies $(\alpha(\sigma), j) \models p^\alpha$ and $(\alpha(\sigma), j) \models q^\alpha$. By the induction hypothesis, this implies that $(\sigma, j) \models p$ and $(\sigma, j) \models q$, leading to $(\sigma, j) \models p \land q$, therefore, $(\sigma, j) \models \psi$. The case that $\psi = p \lor q$ is treated in a similar way.

- Next, consider the case that $\psi = \square p$. Since $(\square p)^\alpha = \square (p^\alpha)$, $(\alpha(\sigma), j) \models \psi^\alpha$ implies $(\alpha(\sigma), k) \models p^\alpha$, for every $k \geq j$. By the induction hypothesis, it follows that $(\sigma, k) \models p$, for every $k \geq j$ and, therefore, $(\sigma, j) \models \square p = \psi$. The case of the other temporal operatos is treated similarly.
Formulas not in Positive Form

Next, consider the case that $\psi$ is not necessarily in positive form. Assume that $\psi$ contains the maximal sub-assertions $p_1, \ldots, p_m$ under positive polarity and the maximal sub-assertions $q_1, \ldots, q_n$ under negative polarity. Let $\varphi = \text{pos}(\psi)$ be the positive-form formula congruent to $\psi$. Formula $\varphi$ can be obtained from $\psi$ by repeatedly applying the following rewrite transformations until none is applicable anymore:

\[
\begin{align*}
\neg(p \lor q) & \rightarrow (\neg p) \land (\neg q) \\
\neg(p \land q) & \rightarrow (\neg p) \lor (\neg q) \\
\neg
\neg p & \rightarrow p \\
\neg\Box p & \rightarrow \Box \neg p \\
\neg\square p & \rightarrow \Diamond \neg p \\
\neg\Diamond p & \rightarrow \lozenge \neg p \\
\neg(p \cup q) & \rightarrow (\neg p) \lor (\neg q) \\
\neg(p \lor q) & \rightarrow (\neg p) \lor (\neg q)
\end{align*}
\]

It is not difficult to see that $\varphi$ will have the form $\varphi(p_1, \ldots, p_m, \neg q_1, \ldots, \neg q_n)$, where $p_1, \ldots, p_m, q_1, \ldots, q_n$ are the maximal sub-assertions occurring in $\psi$. 
Proof Continued

Obviously,

\[
\psi(p_1, \ldots, p_m, q_1, \ldots, q_n) \approx \varphi(p_1, \ldots, p_m, \neg q_1, \ldots, \neg q_n)
\]

Assume that \((\alpha(\sigma), j) \models \psi^\alpha\). Applying the recipe for abstracting a temporal property, we get \(\psi^\alpha = \psi(\alpha^-(p_1), \ldots, \alpha^-(p_m), \alpha^+(q_1), \ldots, \alpha^+(q_n))\). In comparison, applying the same recipe to the positive form formula \(\varphi\) yields \(\varphi^\alpha = \varphi(\alpha^-(p_1), \ldots, \alpha^-(p_m), \alpha^-(\neg q_1), \ldots, \alpha^-(\neg q_n))\).

If we could establish that \(\alpha^-(\neg q_k)\) is congruent to \(\alpha^+(q_k)\), for every \(k = 1, \ldots, n\), we would establish that \(\psi^\alpha\) is congruent to \(\varphi^\alpha\). In general, \(\alpha^-(\neg q_k)\) is not congruent to \(\alpha^+(q_k)\). However, the two are congruent over all abstract states \(S_i\) which have at least one \(\alpha\)-source. We conclude that \(\psi^\alpha\) is congruent to \(\varphi^\alpha\) over all sequences of the form \(\alpha(\sigma)\).

It follows that \(((\alpha(\sigma), j) \models \psi^\alpha\) implies \(((\alpha(\sigma), j) \models \varphi^\alpha\). Since \(\varphi\) is in positive form, this implies \((\sigma, j) \models \varphi\), from which we can conclude \((\sigma, j) \models \psi\)
And Now to Systems

Given an FDS $D = \langle V, O, W, \Theta, \rho, J, C \rangle$, there exists a temporal formula $Sem(D)$, called the temporal semantics of $D$, such that, for every infinite state sequence $\sigma$,

$$\sigma \models Sem(D) \quad \text{iff} \quad \sigma \text{ is a computation of } D.$$

$Sem(D)$ is given by:

$$\Theta \land \Box \rho(V, \bigcirc V) \land \bigwedge_{J \in J} \Box \Diamond J \land \bigwedge_{(p,q) \in C} (\Box \Diamond p \rightarrow \Box \Diamond q)$$

Given a verification problem $D \models \psi$, we construct the temporal formula

$$Ver(D, \psi): \quad Sem(D) \rightarrow \psi.$$

It is not difficult to establish that $D \models \psi$ iff $Ver(D, \psi)$ is valid.
Sound Joint Abstraction

For an FDS $\mathcal{D} = \langle V, \mathcal{O}, W, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$, we define the $\alpha$-abstracted version of $\mathcal{D}$ to be the FDS $\mathcal{D}^\alpha = \langle V_A, \mathcal{O}_A, W_A, \Theta^\alpha, \rho^\alpha, \mathcal{J}^\alpha, \mathcal{C}^\alpha \rangle$, where

$$
\begin{align*}
\Theta^\alpha & = \alpha^+(\Theta) \\
\rho^\alpha & = \alpha^{++}(\rho) \\
\mathcal{J}^\alpha & = \{ \alpha^+(J) \mid J \in \mathcal{J} \} \\
\mathcal{C}^\alpha & = \{ (\alpha^-(p), \alpha^+(q)) \mid (p, q) \in \mathcal{C} \}
\end{align*}
$$

Where,

$$\alpha^{++}(\rho) = \exists V, V' \left( \forall V_A = \mathcal{E}^\alpha(V) \land V'_A = \mathcal{E}^\alpha(V') \land \rho(V, V') \right)$$

Soundness:

If $\mathcal{D}$ and $\psi$ are abstracted according to the recipe presented above, then

$$\mathcal{D}^\alpha \models \psi^\alpha \quad \text{implies} \quad \mathcal{D} \models \psi.$$
Predicate Abstraction

An important question is how to choose the abstraction mapping $\alpha = \mathcal{E}_\alpha : \Sigma \mapsto \Sigma_A$ in a way that will lead to a correct proof. One partial answer is provided by the method of predicate abstraction.

Let $p_1, p_2, \ldots, p_k$ be the set of all atomic formulas referring to the data (non-control) variables appearing within conditions in the program $P$ and within the temporal formula $\psi$.

Following [BBM95], define abstract boolean variables $B_{p_1}, B_{p_2}, \ldots, B_{p_k}$, one for each atomic data formula. The abstraction mapping $\alpha$ is defined by

$$\alpha: \{ B_{p_1} = p_1, B_{p_2} = p_2, \ldots, B_{p_k} = p_k \}$$
Example: Program BAKERY-2

local \( y_1, y_2 \) : natural initially \( y_1 = y_2 = 0 \)

\[
P_1 ::
\begin{align*}
l_0 & : \text{loop forever do} \\
& \begin{cases}
l_1 & : \text{Non-Critical} \\
l_2 & : y_1 := y_2 + 1 \\
l_3 & : \text{await} \ y_2 = 0 \lor y_1 < y_2 \\
l_4 & : \text{Critical} \\
l_5 & : y_1 := 0
\end{cases}
\end{align*}
\]

\[
P_2 ::
\begin{align*}
m_0 & : \text{loop forever do} \\
& \begin{cases}
m_1 & : \text{Non-Critical} \\
m_2 & : y_2 := y_1 + 1 \\
m_3 & : \text{await} \ y_1 = 0 \lor y_2 \leq y_1 \\
m_4 & : \text{Critical} \\
m_5 & : y_2 := 0
\end{cases}
\end{align*}
\]

The temporal properties for program BAKERY-2 are

\[
\psi_{exc} : \Box \neg (\text{at}_l_4 \land \text{at}_m_4)
\]

\[
\psi_{acc} : \Box (\text{at}_l_2 \rightarrow \Diamond \text{at}_l_4)
\]
Abstracting Program BAKERY-2

Define abstract variables $B_{y_1=0}$, $B_{y_2=0}$, and $B_{y_1<y_2}$.

local $B_{y_1=0}, B_{y_2=0}, B_{y_1<y_2} : boolean$

initially $B_{y_1=0} = B_{y_2=0} = 1, B_{y_1<y_2} = 0$

\[
\begin{align*}
\ell_0 & : \text{loop forever do} \\
\ell_1 & : \text{Non-Critical} \\
\ell_2 & : (B_{y_1=0}, B_{y_1<y_2}) := (0, 0) \\
\ell_3 & : \text{await } B_{y_2=0} \lor B_{y_1<y_2} \\
\ell_4 & : \text{Critical} \\
\ell_5 & : (B_{y_1=0}, B_{y_1<y_2}) := (1, \neg B_{y_2=0})
\end{align*}
\]

The abstracted properties can now be model-checked.
Abstraction Alone is Insufficient

Not all properties can be proven by pure finitary abstraction. Consider the program LOOP.

\[
\begin{align*}
 y &: \text{ natural} \\
 \ell_0 &: \text{ while } y > 0 \text{ do} \\
 & \quad \begin{cases} 
 \ell_1 &: \ y := y - 1 \\
 \ell_2 &: \ 	ext{skip} 
\end{cases} \\
 \ell_3 &: 
\end{align*}
\]

Termination of this program cannot be proven by pure finitary abstraction. For example, the abstraction \( \alpha : \mathbb{N} \mapsto \{0, 1\} \) leads to the abstracted program

\[
\begin{align*}
 Y &: \{0, 1\} \\
 \ell_0 &: \text{ while } Y = 1 \text{ do} \\
 & \quad \begin{cases} 
 \ell_1 &: \ Y := \text{sub1}(Y) \\
 \ell_2 &: \ 	ext{skip} 
\end{cases} \\
 \ell_3 &: 
\end{align*}
\]

where

\[
\text{sub1}(Y) = \begin{cases} 
 \{0, 1\} & \text{if } Y = 1 \\
 \{0\} & \text{else}
\end{cases}
\]

This abstracted program may diverge!
The Solution: Augment the Program with a Non-Constraining
Progress Monitor

\[ y : \text{natural} \]

\[
\begin{align*}
\ell_0 : \text{while } y > 0 \text{ do} & \quad \ell_3 : \\
\ell_1 : y := y - 1 & \quad \ell_2 : \text{skip} \\
\end{align*}
\]

\[
\begin{align*}
\text{always do} & \quad m_0 : inc := \text{sign}(y' - y) \\
\text{compassion} (inc < 0, inc > 0) & \quad inc : \{-1, 0, 1\}
\end{align*}
\]

Forming the cross product, we obtain:

\[
\begin{align*}
y & : \text{natural} \\
inc & : \{-1, 0, 1\} \\
\text{compassion} (inc < 0, inc > 0)
\end{align*}
\]

\[
\begin{align*}
\ell_0 : \text{while } y > 0 \text{ do} & \quad \ell_3 : \\
\ell_1 : (y, inc) := (y - 1, \text{sign}(y' - y)) & \quad \ell_2 : inc := \text{sign}(y' - y)
\end{align*}
\]
Abstracting the Augmented System

We obtain the program

\[
\begin{align*}
Y & : \{0, 1\} \\
inc & : \{-1, 0, 1\} \\
compassion & : (inc < 0, inc > 0)
\end{align*}
\]

\(\ell_0\) : while \(Y = 1\) do

\[
\begin{align*}
\ell_1 : (Y, inc) & := \begin{cases} 
\text{if } Y = 1 & (1, 0, -1) \\
\text{then } & (0, 0) \\
\text{else } & \end{cases} \\
\ell_2 : inc & := 0 \\
\ell_3 & :
\end{align*}
\]

Which always terminate.
A More Complicated Case

Sometimes we need a more complex progress measure:

\[
\begin{align*}
y &\text{ natural} \\
\ell_0 &\text{ while } y > 1 \text{ do} \\
\ell_1 &\quad y := y - 2 \\
\ell_2 &\quad y := \{y + 1, y\} \\
\ell_3 &\quad \text{skip} \\
\ell_4 &
\end{align*}
\]

To prove termination of this program we augment it by the monitor:

\[
\begin{align*}
\text{define} &\quad \delta = y + \text{at } \ell_2 \\
\text{inc} &\quad : \{-1, 0, 1\} \\
\text{compassion} &\quad (\text{inc} < 0, \text{inc} > 0) \\
m_0 &\quad \text{always do} \\
&\quad \text{inc} := \text{sign}(\delta' - \delta)
\end{align*}
\]
Complicated Case Continued

Augmenting and abstracting, we get:

\[
\begin{align*}
Y & : \{0, \text{one, large}\} \\
inc & : \{-1, 0, 1\} \\
\text{compassion} & : (inc < 0, inc > 0)
\end{align*}
\]

\[\ell_0 : \text{while } Y = \text{large} \text{ do} \]

\[
\begin{aligned}
\ell_1 : (Y, inc) & := (\text{sub2}(Y), -1) \\
\ell_2 : (Y, inc) & := (\{\text{add1}(Y), Y\}, \{0, -1\}) \\
\ell_3 : inc & := 0
\end{aligned}
\]

\[\ell_4 : \]

where,

\[
\begin{align*}
\text{sub2}(Y) & = \\
& \text{if } Y \in \{0, \text{one}\} \text{ then } 0 \text{ else } \{0, \text{one, large}\}
\end{align*}
\]

\[
\begin{align*}
\text{add1}(Y) & = \text{if } Y = 0 \text{ then one else large}
\end{align*}
\]

This program always terminates
Verification by Augmented Finitary Abstraction - The VAA Method

To verify that $\psi$ is $D$-valid,

- Optionally choose a non-constraining progress monitor $FDS M$ and let $\mathcal{A} = D \parallel M$. In case this step is skipped, we let $\mathcal{A} = D$.

- Choose a finitary state abstraction mapping $\alpha$ and calculate $\mathcal{A}^\alpha$ and $\psi^\alpha$ according to the sound recipes.

- Model check $\mathcal{A}^\alpha \models \psi^\alpha$.

- Infer $D \models \psi$.

**Claim 13.** The VAA method is complete, relative to deductive verification.

That is, whenever there exists a deductive proof of $D \models \psi$, we can find a finitary abstraction mapping $\alpha$ and a non-constraining progress monitor $M$, such that $\mathcal{A}^\alpha \models \psi^\alpha$. 