This abstracted program may diverge:

\[
\text{pos = } \lambda \text{ pos then zero, else zero = } \lambda \text{ sub1}
\]

where

\[
\begin{align*}
\text{skip} & : \mathbb{Z} \\
(\lambda \text{ sub1} = : \mathbb{Z} & : \mathbb{Z}) \\
\text{while} \text{ pos = } & \lambda \text{ : 0} \text{ sub1} \\
\{\text{pos, zero}\} & : \lambda
\end{align*}
\]

Termination of this program cannot be proven by pure finitary abstraction. For example, the abstraction \( \mathbb{N} \leftarrow \{\text{pos, zero}\} \) leads to the abstracted program:

\[
\begin{align*}
\text{skip} & : \mathbb{Z} \\
1 - \text{ i} & = : \mathbb{Z} \\
\text{while} \text{ 0 < i do} & : 0 \text{ sub1} \\
\text{natural} & : \mathbb{N}
\end{align*}
\]

Consider the program loop.

State abstraction alone is insufficient.
Solution: Augmentation with a Non-Constraining Progress Monitor

\[

c\cdot \text{sign}(\hat{r} - r) \cdot \text{inc} : \emptyset
\]

\[

\text{always do } (0 < \text{inc} > 0, \text{inc}) \text{ compassion } \{1,0\} : \text{inc} \text{ natural}
\]

\[

\text{while } \text{do } (0 < r, \text{inc}) \text{ compassion } \{1,0\} : \text{inc} \text{ natural}
\]

\[

\text{while } \text{do } 0 < r : 0
\]
Which always terminates.

\[
\begin{align*}
0 & =: \text{inc} : \text{J} \\
(0, \text{zero}) & \quad \text{else} \\
(1, \{\text{pos, zero}\}) & \quad \text{then} \\
\text{pos} = \lambda & \quad \text{if} \\
\text{while pos} = \lambda \text{ do} & \\
0 & : \text{J} \\
(0 < \text{inc}) & \quad \text{compassion} \\
\{1, 0\} & : \text{inc} \\
\{\text{pos, zero}\} & : \lambda
\end{align*}
\]

We obtain the program:

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

\[ \text{inc} := \text{sign}(\theta_i - \theta) \]

always do

\[(0 < \text{inc}, 0 > \text{inc}) \]

\{(\theta_i' > 0, 1 - \theta_i') : inc \}

\(z + \theta_i = 0 \)

define

Sometimes we need a more complex progress measure:

A More Complicated Case
Augmenting and abstracting, we get:

```
0 < inc 0 > inc
{ I, 0 } : inc
{ I, 0, one, large } : λ
```

This program always terminates.

```
if 0 then zero else large
if 0 then one else large
```

```
add1 = \( \lambda \)
sub2 = \( \lambda \)
```

where:

```
0 = \( \lambda \)
(1 - I) = \( \lambda \)
```

```
while λ while large = large do
0 < inc 0 > inc
{ I, 0 } : inc
{ I, 0, one, large } : λ
```

Latin: Augmenting and abstracting, we get:

```
Complicated Case Continued
```

A. Pnueli

Advanced Topics in Reactive Verification, NYU, Spring, 2004
A. Pnueli

Abstraction for Liveness

Verification by Augmented Finitary Abstraction - The VAA Method

To verify that $4$ is $5$-valid, optionally choose a non-constraining progress monitor $FDS$ and let $M$. In case this step is skipped, we let $M = A$. Optionally choose a non-constraining progress monitor $FDS$ and let

\[ A \upharpoonright \quad M \quad \upharpoonright \quad A = A \]

\[ \forall \phi \quad \exists \psi \]

\[ \phi \models A \psi \]

Claim 2. The VAA method is complete, relative to deductive verification.

Model check $\forall A \phi$. Choose a finitary state abstraction mapping $\alpha$ and calculate $A \alpha \phi$ and $A \alpha \phi$. According to the sound recipes.

Infer $\phi$. Let $M_\alpha \models \phi$. Such that $\forall \phi$ we can find a finitary abstraction mapping $\alpha$ and a non-constraining progress monitor $M$, such that Whenever there exists a deductive proof of $\phi \models \alpha$, we can find a finitary abstraction for $\alpha$, and a non-constraining progress monitor $M$, such that $\forall \phi$ we can find a finitary abstraction mapping $\alpha$ and a non-constraining progress monitor $M$, such that $\forall \phi$ we can find a finitary abstraction mapping $\alpha$ and a non-constraining progress monitor $M$, such that $\forall \phi$ we can find a finitary abstraction mapping $\alpha$ and a non-constraining progress monitor $M$, such that $\forall \phi$ we can find a finitary abstraction mapping $\alpha$ and a non-constraining progress monitor $M$, such that
Some Comments

One possible interpretation of our results is that it is not sufficient to abstract states. One should also abstract transitions.

Our completeness results can also be related to the work of Abadi and Lamport [AL91] who claim that abstraction can be made complete by the addition of history and prophecy variables. In our case, the history is represented by the reference and prophecy variables. The prophecy is represented by the compass requirement.