Recall Compositional Recurrence Analysis

• $D$: set of arithmetic transition formulas

• $\varphi \otimes \psi \triangleq \exists x''. \varphi[x' \mapsto x''] \land \psi[x \mapsto x'']$

• $\varphi \oplus \psi \triangleq \varphi \lor \psi$

• $\varphi^* \sim$ extract + solve system of recurrence relations
Tensor semantic algebra of CRA

Tensored transition formula \( \sim \) formula over \textit{four} copies of the program variables

\[ x, x', \bar{x}, \bar{x}' \]
Tensor semantic algebra of CRA

Tensored transition formula \( \sim \) formula over \textit{four} copies of the program variables

\( x, x', \overline{x}, \overline{x}' \)

Beginning of path

End of path

\( \phi_t \equiv \phi[x_7!; x_7!; x'_{7!}; x'_{7!}] \)

\( \phi \odot \equiv \phi[x_7!; x_7!; x'_{7!}; x'_{7!}] \)

\( \text{readout} \equiv \phi[m_7!; m_7!; m_7!; m_7!; x_7!; x_7!] \)

\( \text{readout}(x' = x + 1) \odot (x' = 2x) = (x' = x + 1) \odot (x' = 2x) \)
Tensor semantic algebra of CRA

Tensored transition formula over four copies of the program variables

Beginning of path

Beginning of continuation

End of continuation

End of path

\[ \begin{align*}
\phi_t & \equiv \phi \left[ x_7; x_7 \right] \\
\phi \otimes & \equiv \phi \left[ x_7; x_7 \right] \cdot \left[ x_7; x_7 \right]
\end{align*} \]

E.g., \((x' = x + 1) \otimes (x' = 2x) = (x' = x + 1)^{x' = 2x}\)

\[ \text{readout} \equiv 9m : \left[ x_7; x_7 ; x_7; x_7 \right] \]

\[ \text{readout} \left[ x_7; x_7 ; x_7; x_7 \right] = x_7; x_7 + 2\]
Tensor semantic algebra of CRA

Tensored transition formula $\sim$ formula over *four* copies of the program variables

$$x, x', \overline{x}, \overline{x'}$$

- $\Phi \otimes \Psi \triangleq \exists x'', \overline{x}''. \Phi[x \mapsto x'', \overline{x} \mapsto \overline{x}''] \land \Psi[x' \mapsto x'', \overline{x'} \mapsto \overline{x}'']$
Tensor semantic algebra of CRA

Tensored transition formula \(\sim\) formula over \textit{four} copies of the program variables

\[ x, x', \overline{x}, \overline{x}' \]

- \(\Phi \otimes \Psi \triangleq \exists x'', \overline{x}''. \Phi [x \mapsto x'', \overline{x} \mapsto \overline{x}''] \land \Psi [x' \mapsto x'', \overline{x}' \mapsto \overline{x}'']\)
- \(\Phi \oplus \Psi, \Phi^*\) as for the untensored case
Tensor semantic algebra of CRA

Tensored transition formula $\sim$ formula over *four* copies of the program variables

$$x, x', \overline{x}, \overline{x}$$

- $\Phi \otimes \Psi \triangleq \exists x'', \overline{x}''. \Phi[x \mapsto x'', \overline{x} \mapsto \overline{x}''] \land \Psi[\overline{x}' \mapsto \overline{x}'', x' \mapsto x'']$
- $\Phi \oplus \Psi, \Phi^*$ as for the untensored case
- $\varphi^t \triangleq \varphi[x \mapsto x', \overline{x}' \mapsto x]$
Tensor semantic algebra of CRA

Tensored transition formula $\sim$ formula over \textit{four} copies of the program variables

$x, x', \overline{x}, \overline{x'}$

- $\Phi \otimes \Psi \triangleq \exists x'', \overline{x}''. \Phi[x \mapsto x'', \overline{x} \mapsto \overline{x}''] \land \Psi[x' \mapsto x'', \overline{x'} \mapsto \overline{x''}]$
- $\Phi \oplus \Psi, \Phi^*$ as for the untensored case
- $\varphi^t \triangleq \varphi[x \mapsto x', x' \mapsto x]$
- $\varphi \odot \psi \triangleq \varphi[x \mapsto \overline{x}, x' \mapsto \overline{x'}] \land \psi[x \mapsto \overline{x}, x' \mapsto \overline{x'}]$
  - E.g., $(x' = x + 1) \odot (x' = 2x) = (x' = x + 1 \land \overline{x'} = 2 \overline{x})$
Tensor semantic algebra of CRA

Tensored transition formula $\sim$ formula over *four* copies of the program variables

$$x, x', \overline{x}, \overline{x}'$$

- $\Phi \otimes \Psi \triangleq \exists x'', \overline{x}''. \Phi[x \mapsto x'', \overline{x} \mapsto \overline{x}''] \land \Psi[x' \mapsto x'', \overline{x}' \mapsto \overline{x}''']$
- $\Phi \oplus \Psi, \Phi^*$ as for the untensored case
- $\varphi^t \triangleq \varphi[x \mapsto x', x' \mapsto x]$
- $\varphi \odot \psi \triangleq \varphi[x \mapsto \overline{x}, x' \mapsto \overline{x}'] \land \psi[x \mapsto \overline{x}, x' \mapsto \overline{x}']$
  - E.g., $(x' = x + 1) \odot (x' = 2x) = (x' = x + 1 \land \overline{x}' = 2\overline{x})$
- $\text{readout}(\Phi) \triangleq \exists m. \Phi[x \mapsto x, x' \mapsto m, \overline{x} \mapsto \overline{m}, \overline{x}' \mapsto \overline{x}']$
  - E.g., $\text{readout}(x' = \overline{x} + 1 \land \overline{x}' = 2\overline{x}) = \overline{x}' = 2\overline{x} + 2$
Newtonian program analysis is a nested fixpoint computation

\[ \vec{v}^{(0)} = \vec{f}(0) \]
\[ \vec{v}^{(i+1)} = \vec{Y}^{(i)} \]

where \( \vec{Y}^{(i)} \) is the least solution of

\[ \vec{Y} = \vec{f}(\vec{v}^{(i)}) \oplus D_{\vec{f}} \big|_{\vec{v}^{(i)}} (\vec{Y}) \]
Newtonian program analysis is a nested fixpoint computation

\[ \vec{v}^{(0)} = \vec{f}(\vec{0}) \]
\[ \vec{v}^{(i+1)} = \vec{Y}^{(i)} \]

where \( \vec{Y}^{(i)} \) is the least solution of

\[ \vec{Y} = \vec{f}(\vec{v}^{(i)}) \oplus D\vec{f}_{\mid \vec{v}^{(i)}}(\vec{Y}) \]

**Outer fixpoint computation** requires two ingredients:

1. **Ascending chain condition**
   - \( p_0 \leq p_1 \leq p_2 \leq \ldots \) eventually stabilizes

2. **Decidable entailment**
   - Need to be able to check \( p_{i+1} \leq p_i \)
Problem: (CRA) transition formulas have neither

1. Transition formulas have infinite ascending chains (convergence is not guaranteed)
2. Transition formula entailment is undecidable (convergence can’t be detected)
Problem: (CRA) transition formulas have neither

1. Transition formulas have infinite ascending chains (convergence is not guaranteed)
2. Transition formula entailment is undecidable (convergence can’t be detected)

Solution: outer fixpoint computation in a domain with decidable equivalence + widening.
Problem: (CRA) transition formulas have neither

1. Transition formulas have infinite ascending chains (convergence is not guaranteed)
2. Transition formula entailment is undecidable (convergence can’t be detected)

Solution: outer fixpoint computation in a domain with decidable equivalence + widening.

Incur precision loss due to abstraction + widening
Problem: (CRA) transition formulas have neither

1. Transition formulas have infinite ascending chains (convergence is not guaranteed)
2. Transition formula entailment is undecidable (convergence can’t be detected)

Solution: outer fixpoint computation in a domain with decidable equivalence + widening.
Incur precision loss due to abstraction + widening

*We can do better!*
CRA’s iteration operator

```python
while (i < n):
    if (*):
        x := x + i
    else:
        y := y + i
    i := i + 1
```

Loop body:

\[
i \leq n \\
\land \left( (x' = x + i \land y' = y) \lor (y' = y + i \land x' = x) \right) \\
\land i' = i + 1 \\
\land n' = n
\]

Recurrences:

\[
i^{(k)} = i^{(k-1)} + 1 \\
x^{(k)} + y^{(k)} = x^{(k-1)} + y^{(k-1)} + i \\
x^{(k)} \geq x^{(k-1)} \\
y^{(k)} \geq y^{(k-1)}
\]

Loop abstraction:

\[
\exists k. k \geq 0 \land i' = i + k \land x' + y' = x + y + k(k+1)/2 + ki_0 \land x' \geq x \land y' \geq y
\]
CRA’s iteration operator

while \(i < n\):
    if (*):
        \(x := x + i\)
    else
        \(y := y + i\)
    \(i := i + 1\)

\(i \leq n\)
\(\land (x' = x + i \land y' = y)\)
\(\lor (y' = y + i \land x' = x)\)
\(\land i' = i + 1\)
\(\land n' = n\)

\(\alpha\)

Polyhedron

\(i^{(k)} = i^{(k-1)} + 1\)
\(x^{(k)} + y^{(k)} = x^{(k-1)} + y^{(k-1)} + i\)
\(x^{(k)} \geq x^{(k-1)}\)
\(y^{(k)} \geq y^{(k-1)}\)

\(\forall \alpha. k \geq 0 \land i' = i + k \land x' + y' = x + y + k(k+1)/2 + ki_0 \land x' \geq x \land y' \geq y\)
**Iteration domains**

[Kincaid, Breck, Boroujeni, Reps ’17]

\[ \varphi^* = cl(\alpha(\varphi)) \]

**Key idea:** we have an opportunity to detect / enforce convergence at every place we apply the \( \ast \) operator.
1. Rewrite system of equations so all variables appear below a star (\(\sim\) Gauss-Jordan elimination):

\[ X = aXbXc + d \sim X = d \times (a \odot bXc)^* \]

Detensor product \(\approx\) readout
1 Rewrite system of equations so all variables appear below a star (∼ Gauss-Jordan elimination):

\[ X = aXbXc + d \leadsto X = d \star (a \odot bXc)^* \]

2 Resulting system can be solved iteratively:

\[ \nu_0 = 0 \]
\[ \nu_1 = d \star cl(p_0) \]
\[ \vdots \]

\[ p_0 = \alpha(a \odot b\nu_0 c) \]
\[ p_1 = p_0 \triangledown \alpha(a \odot b\nu_1 c) \]
\[ \vdots \]

(repeat until \( p_{n+1} = p_n \))
Algebraic analyses can be extended to recursive procedures using

1. *Tensor domains*, to re-arrange recursion into loops
2. *Iteration domains*, to detect and enforce convergence