Recall Compositional Recurrence Analysis

- D: set of arithmetic *transition formulas*
- $\varphi \otimes \psi \triangleq \exists \mathbf{x}''. \varphi[\mathbf{x}' \mapsto \mathbf{x}''] \land \psi[\mathbf{x} \mapsto \mathbf{x}'']$
- $\varphi \oplus \psi \triangleq \varphi \lor \psi$
- $\varphi^* \sim$ extract + solve system of recurrence relations

PRINCETON UNIVERSITY WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

Tensor semantic algebra of CRA

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

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

Tensor semantic algebra of CRA

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

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

Beginning of path

End of path

PRINCETON UNIVERSITY WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

Tensor semantic algebra of CRA

Tensor … *four* copies of the program variables

Beginning of continuation

End of continuation

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

Beginning of path

End of path

PRINCETON UNIVERSITY

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Tensor semantic algebra of CRA

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

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

- $\Phi \otimes \Psi \triangleq \exists \underline{\mathbf{x}}'', \overline{\mathbf{x}}''. \Phi[\underline{\mathbf{x}} \mapsto \underline{\mathbf{x}}'', \overline{\mathbf{x}} \mapsto \overline{\mathbf{x}}''] \wedge \Psi[\underline{\mathbf{x}}' \mapsto \underline{\mathbf{x}}'', \overline{\mathbf{x}}' \mapsto \overline{\mathbf{x}}'']$

PRINCETON UNIVERSITY WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

Tensor semantic algebra of CRA

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

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

- $\Phi \otimes \Psi \triangleq \exists \underline{\mathbf{x}}'', \overline{\mathbf{x}}''. \Phi[\underline{\mathbf{x}} \mapsto \underline{\mathbf{x}}'', \overline{\mathbf{x}} \mapsto \overline{\mathbf{x}}''] \wedge \Psi[\underline{\mathbf{x}}' \mapsto \underline{\mathbf{x}}'', \overline{\mathbf{x}}' \mapsto \overline{\mathbf{x}}'']$
- $\Phi \oplus \Psi$, Φ^* as for the untensored case

PRINCETON UNIVERSITY WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

Tensor semantic algebra of CRA

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

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

- $\Phi \otimes \Psi \triangleq \exists \underline{\mathbf{x}}'', \overline{\mathbf{x}}''. \Phi[\underline{\mathbf{x}} \mapsto \underline{\mathbf{x}}'', \overline{\mathbf{x}} \mapsto \overline{\mathbf{x}}''] \wedge \Psi[\underline{\mathbf{x}}' \mapsto \underline{\mathbf{x}}'', \overline{\mathbf{x}}' \mapsto \overline{\mathbf{x}}'']$
- $\Phi \oplus \Psi$, Φ^* as for the untensored case
- $\varphi^t \triangleq \varphi[\mathbf{x} \mapsto \mathbf{x}', \mathbf{x}' \mapsto \mathbf{x}]$

Tensor semantic algebra of CRA

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

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

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

PRINCETON UNIVERSITY

WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

Tensor semantic algebra of CRA

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

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

- $\Phi \otimes \Psi \triangleq \exists \underline{\mathbf{x}}'', \overline{\mathbf{x}}''. \Phi[\underline{\mathbf{x}} \mapsto \underline{\mathbf{x}}'', \overline{\mathbf{x}} \mapsto \overline{\mathbf{x}}''] \land \Psi[\underline{\mathbf{x}}' \mapsto \underline{\mathbf{x}}'', \overline{\mathbf{x}}' \mapsto \overline{\mathbf{x}}'']$
- $\Phi \oplus \Psi$, Φ^* as for the untensored case
- $\varphi^t \triangleq \varphi[\mathbf{x} \mapsto \mathbf{x}', \mathbf{x}' \mapsto \mathbf{x}]$
- $\varphi \odot \psi \triangleq \varphi[\mathbf{x} \mapsto \underline{\mathbf{x}}, \mathbf{x}' \mapsto \underline{\mathbf{x}}'] \land \psi[\mathbf{x} \mapsto \overline{\mathbf{x}}, \mathbf{x}' \mapsto \overline{\mathbf{x}}']$
  - E.g., $(x' = x + 1) \odot (x' = 2x) = (\underline{x}' = \underline{x} + 1 \land \overline{x}' = 2\overline{x})$
- *readout*$(\Phi) \triangleq \exists \mathbf{m}. \Phi[\underline{\mathbf{x}} \mapsto \mathbf{x}, \underline{\mathbf{x}}' \mapsto \mathbf{m}, \overline{\mathbf{x}} \mapsto \mathbf{m}, \overline{\mathbf{x}}' \mapsto \mathbf{x}']$
  - E.g., *readout*$(\underline{x}' = \underline{x} + 1 \land \overline{x}' = 2\overline{x}) = \mathsf{x}' = 2\mathsf{x} + 2$

Newtonian program analysis is a nested fixpoint computation

Outer fixpoint computation

Inner fixpoint computation

$$\vec{\nu}^{(0)} = \vec{f}(\vec{0})$$
$$\vec{\nu}^{(i+1)} = \vec{Y}^{(i)}$$

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

$$\vec{Y} = \vec{f}(\vec{\nu}^{(i)}) \oplus D\vec{f}|_{\vec{\nu}^{(i)}}(\vec{Y})$$

Newtonian program analysis is a nested fixpoint computation

Outer fixpoint computation

$$\vec{\nu}^{(0)} = \vec{f}(\vec{0})$$
$$\vec{\nu}^{(i+1)} = \vec{Y}^{(i)}$$

Inner fixpoint computation

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

$$\vec{Y} = \vec{f}(\vec{\nu}^{(i)}) \oplus D\vec{f}|_{\vec{\nu}^{(i)}}(\vec{Y})$$

Outer fixpoint computation requires two ingredients:

1. Ascending chain condition
   - $p_0 \leq p_1 \leq p_2 \leq$ ... 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

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

loop body

$$\mathtt{i} < \mathtt{n} \land \left(\begin{array}{l} (\mathtt{x}' = \mathtt{x} + \mathtt{i} \land \mathtt{y}' = \mathtt{y}) \\ \lor \; (\mathtt{y}' = \mathtt{y} + \mathtt{i} \land \mathtt{x}' = \mathtt{x}) \end{array} \right) \land \mathtt{i}' = \mathtt{i} + 1 \land \mathtt{n}' = \mathtt{n}$$

α

recurrences

$$\begin{aligned} \mathtt{i}^{(k)} &= \mathtt{i}^{(k-1)} + 1 \\ \mathtt{x}^{(k)} + \mathtt{y}^{(k)} &= \mathtt{x}^{(k-1)} + \mathtt{y}^{(k-1)} + \mathtt{i} \\ \mathtt{x}^{(k)} &\geq \mathtt{x}^{(k-1)} \\ \mathtt{y}^{(k)} &\geq \mathtt{y}^{(k-1)} \end{aligned}$$

cl

loop abstraction

$$\exists k. k \geq 0 \land \mathtt{i}' = \mathtt{i} + k \land \mathtt{x}' + \mathtt{y}' = \mathtt{x} + \mathtt{y} + k(k+1)/2 + k\mathtt{i}_0 \land \mathtt{x}' \geq \mathtt{x} \land \mathtt{y}' \geq \mathtt{y}$$

PRINCETON UNIVERSITY WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

CRA's iteration operator

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

loop body

$$\begin{aligned}&\mathtt{i} < \mathtt{n}\\ &\wedge \begin{pmatrix} & (\mathtt{x}' = \mathtt{x} + \mathtt{i} \wedge \mathtt{y}' = \mathtt{y}) \\ \vee & (\mathtt{y}' = \mathtt{y} + \mathtt{i} \wedge \mathtt{x}' = \mathtt{x}) \end{pmatrix}\\ &\wedge \mathtt{i}' = \mathtt{i} + 1\\ &\wedge \mathtt{n}' = \mathtt{n}\end{aligned}$$

α

Polyhedron

recurrences

$$\begin{aligned} \mathtt{i}^{(k)} &= \mathtt{i}^{(k-1)} + 1\\ \mathtt{x}^{(k)} + \mathtt{y}^{(k)} &= \mathtt{x}^{(k-1)} + \mathtt{y}^{(k-1)} + \mathtt{i}\\ \mathtt{x}^{(k)} &\geq \mathtt{x}^{(k-1)}\\ \mathtt{y}^{(k)} &\geq \mathtt{y}^{(k-1)} \end{aligned}$$

cl

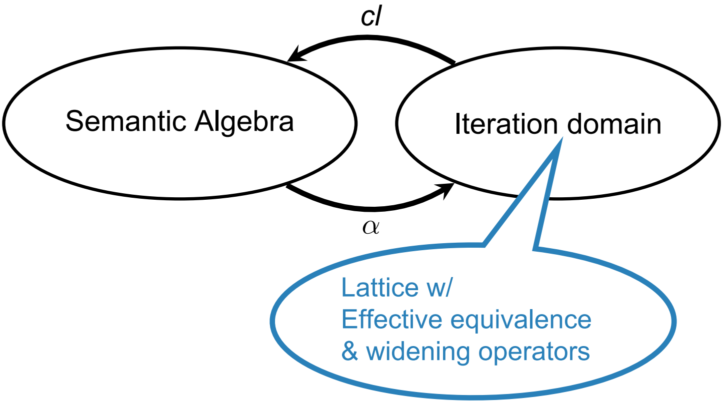
loop abstraction

$$\exists k. k \geq 0 \wedge \mathtt{i}' = \mathtt{i} + k \wedge \mathtt{x}' + \mathtt{y}' = \mathtt{x} + \mathtt{y} + k(k+1)/2 + k\mathtt{i}_0 \wedge \mathtt{x}' \geq \mathtt{x} \wedge \mathtt{y}' \geq \mathtt{y}$$

Iteration domains

[Kincaid, Breck, Boroujeni, Reps ’17]

$$\varphi^* = \mathit{cl}(\alpha(\varphi))$$



Key idea: we have an opportunity to detect / enforce convergence at every place we apply the $*$ operator.

1. Rewrite system of equations so all variables appear below a star ($\sim$ Gauss-Jordan elimination):

$$X = aXbXc + d \leadsto X = d \ltimes (a \odot bXc)^*$$

Detensor product $\approx$ readout

PRINCETON UNIVERSITY WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

1. Rewrite system of equations so all variables appear below a star ($\sim$ Gauss-Jordan elimination):

$$X = aXbXc + d \rightsquigarrow X = d \ltimes (a \odot bXc)^*$$

2. Resulting system can be solved iteratively:

CRA

$$\nu_0 = 0$$

$$\nu_1 = d \ltimes \mathit{cl}(p_0)$$

$$\vdots$$

Polyhedra

$$p_0 = \alpha(a \odot b\nu_0 c)$$

$$p_1 = p_0 \nabla \alpha(a \odot b\nu_1 c)$$

$$\vdots$$

(repeat until $p_{n+1} = p_n$)

PRINCETON UNIVERSITY

WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

Summary

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

PRINCETON UNIVERSITY
WISCONSIN UNIVERSITY OF WISCONSIN–MADISON