Certificate Translation in Abstract Interpretation

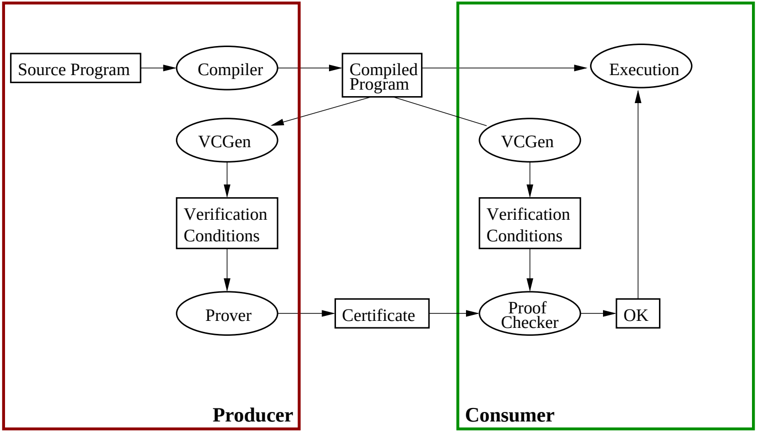
Gilles Barthe and **César Kunz**

Inria

April 2, 2008

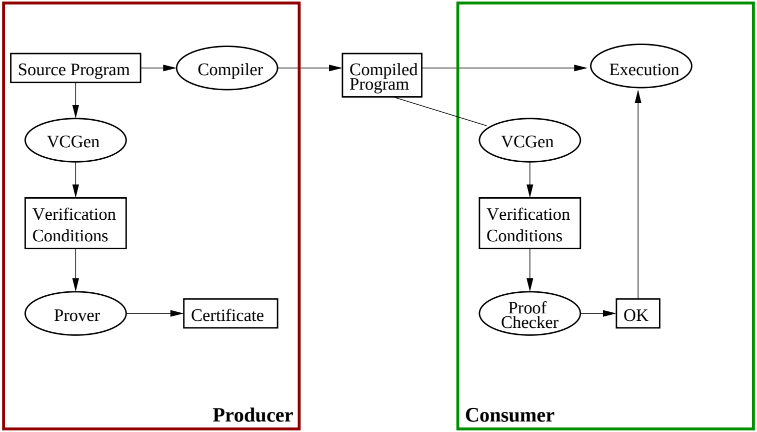
Motivation: source code verification

Traditional PCC



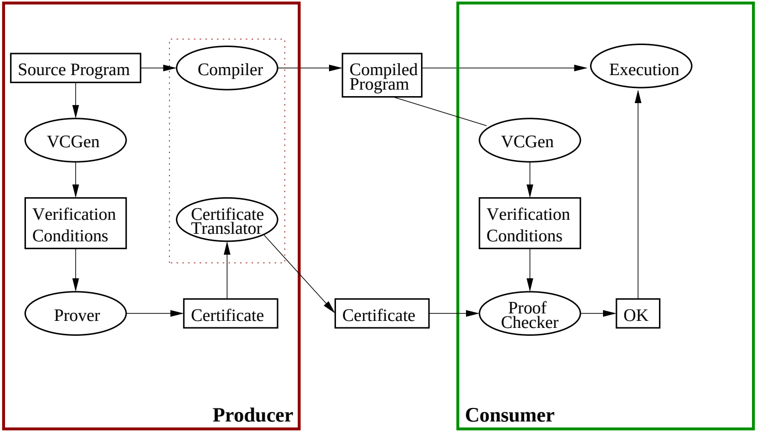
Motivation: source code verification

Source Code Verification

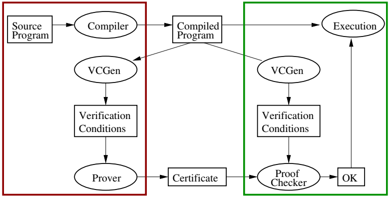


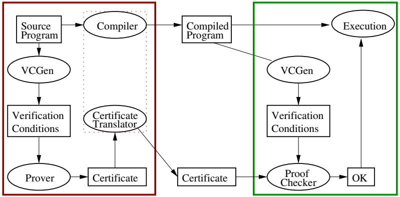
Motivation: source code verification

Certificate Translation



Certificate translation vs certifying compilation





Conventional PCC		Certificate Translation
Automatically inferred invariants	**Specification**	Interactive
Automatic certifying compiler	**Verification**	Interactive source verification
Safety	**Properties**	Complex functional properties

An Abstract Model for Certificate Translation

- particular language
- particular VCgen
- particular program optimizations

} hard to generate a single unifying framework

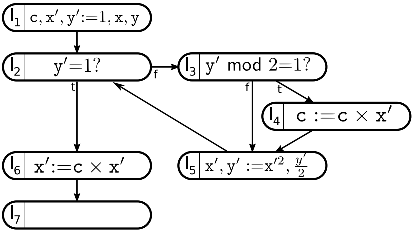
Model: Abstract interpretation of low step trace semantics

1. we show:
   - interactive verification
   - automatic program analysis

   } instances of the same abstract model.
2. study their interaction in certificate translation

Program Representation

```
c := 1
x′ := x
y′ := y
while (y′ ≠ 1) do
   if (y′ mod 2 = 1) then
      c := c × x′
   fi
done
x′ = x′ × c
```

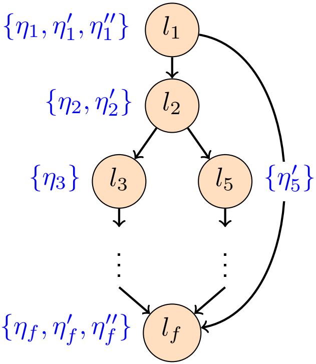


Program: directed graph

- Nodes denoting execution points ($\mathcal{N}$).
- Edges denoting possible transitions between nodes ($\mathcal{E}$).

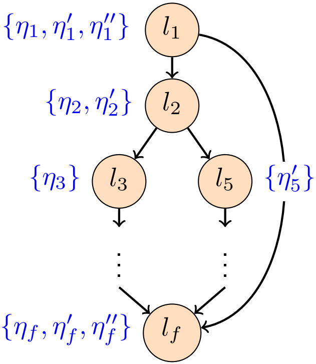
Abstract Interpretation

Program semantics

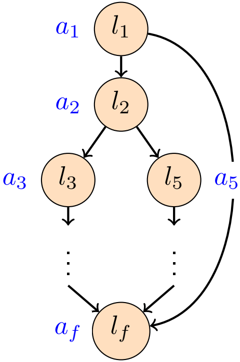


Abstract Interpretation

Program semantics

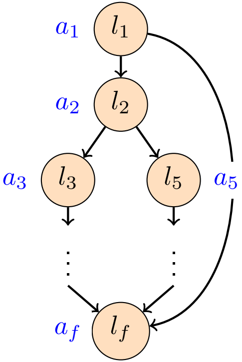


Abstract representation



Solution of an Abstract Interpretation

- $\mathbf{D} = \langle D, \sqsubseteq, \sqcap, \ldots \rangle$,
- $T_{\langle l_i, l_j \rangle} : D \rightarrow D$ a transfer function (for any edge $\langle l_i, l_j \rangle$)

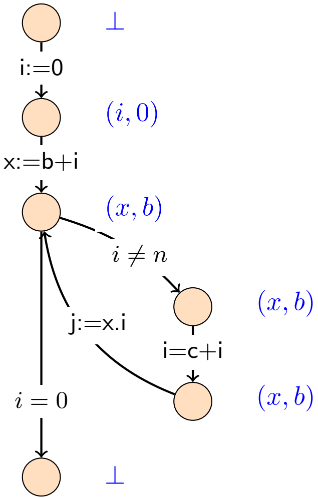


$\{a_1, a_2, \ldots, a_f\}$ a solution of $(\mathbf{D}, T)$ if:

$$T_{\langle l_1, l_2 \rangle}(a_1) \sqsubseteq a_2$$
$$T_{\langle l_2, l_5 \rangle}(a_2) \sqsubseteq a_5$$
$$T_{\langle l_1, l_f \rangle}(a_1) \sqsubseteq a_f$$
$$\ldots$$

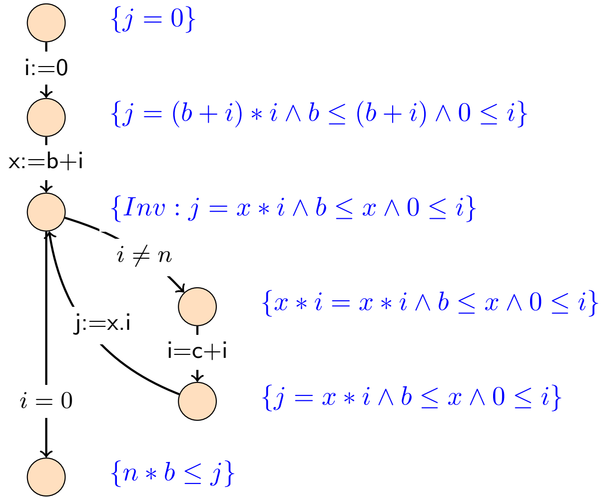
Example of decidable solution (e.g. constant propagation)

(D, T): constant analysis

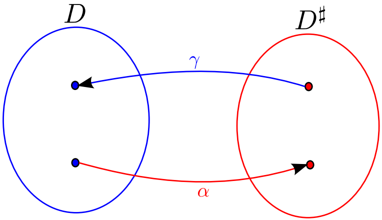


Example of non-decidable solution (e.g. program verification)

(D, T): weakest precondition calculus



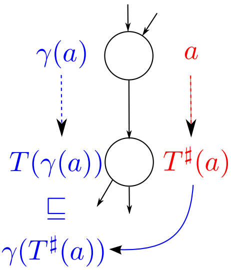
Galois connections captures notion of imprecision



In the following (intuition):

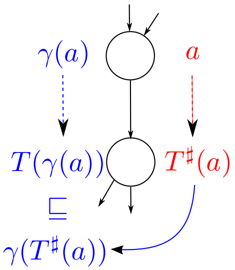
- (D, T): weakest precondition based verification framework
- $(D^\sharp, T^\sharp)$: static analysis that *justifies* a program optimization.

Consistency of $T^\sharp$ w.r.t. T



$$T(\gamma(a)) \sqsubseteq \gamma(T^\sharp(a))$$

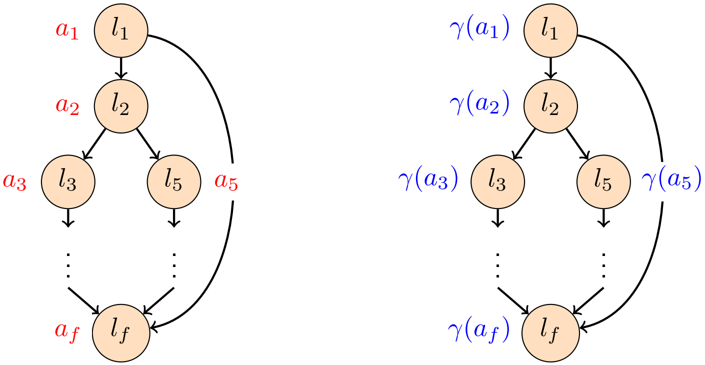
Consistency of $T^\sharp$ w.r.t. T



$$T(\gamma(a)) \sqsubseteq \gamma(T^\sharp(a))$$

Smaller elements: more information

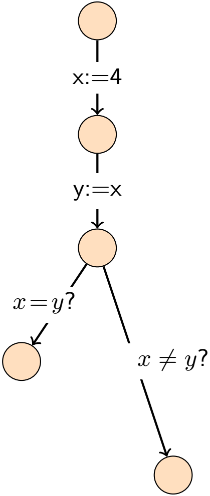
Consistency of $T^\sharp$ w.r.t. T



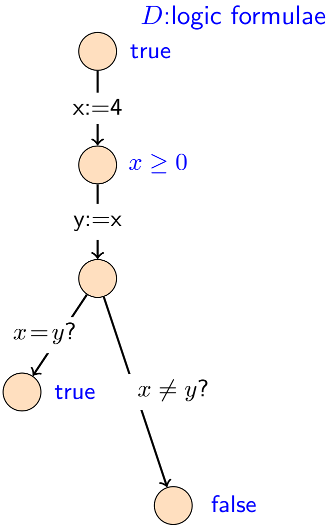
Result:

$\{a_1, a_2 \ldots a_n\}$ a solution of $(D^\sharp, T^\sharp)$, then $\{\gamma(a_1), \gamma(a_2) \ldots \gamma(a_n)\}$ is a solution of (D, T).

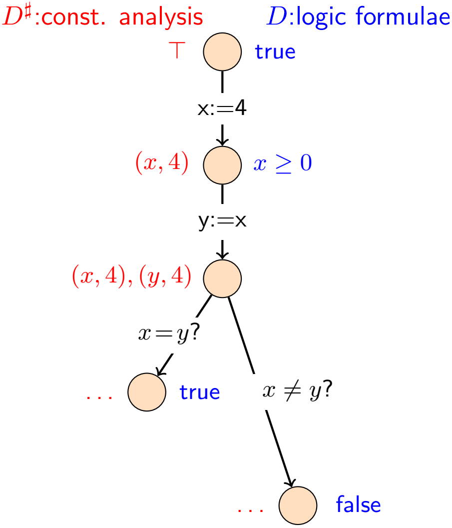
A Primer on Certificate Translation



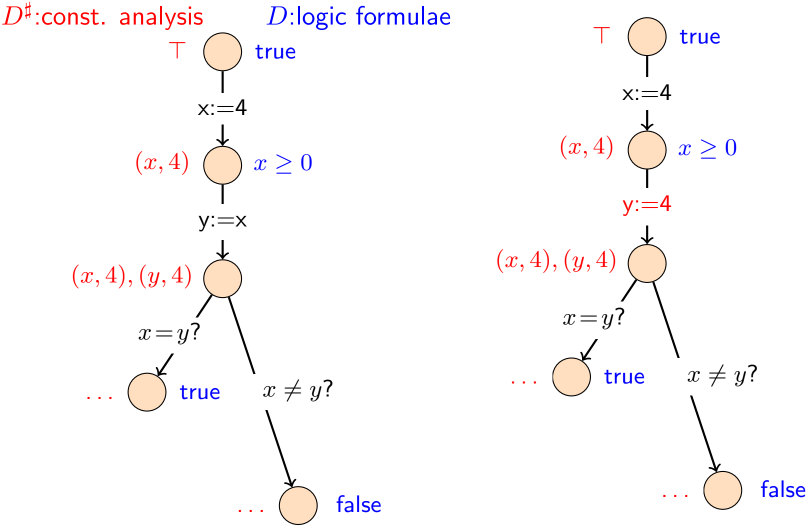
A Primer on Certificate Translation



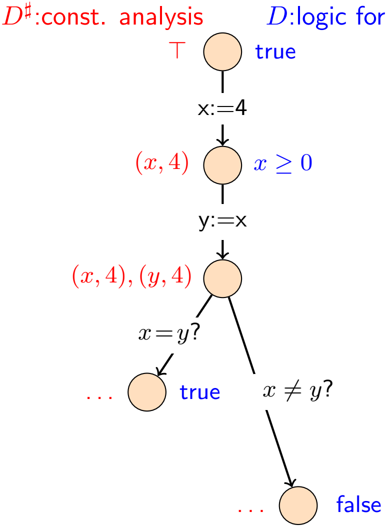
A Primer on Certificate Translation

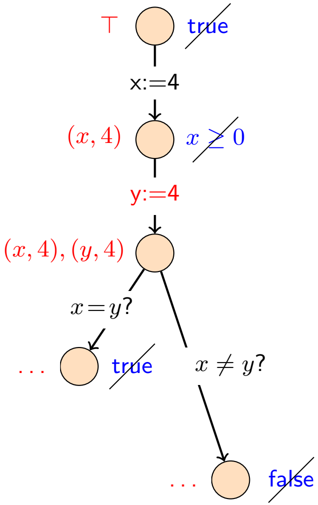


A Primer on Certificate Translation



A Primer on Certificate Translation

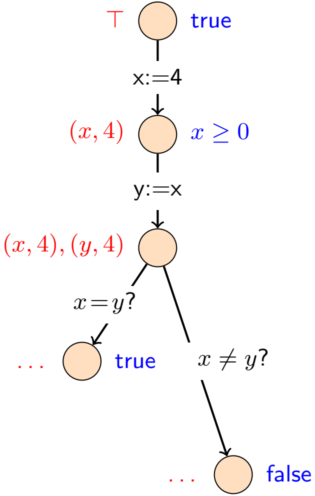




A Primer on Certificate Translation

Key Idea

sufficiently strong solution ↔ preservation along transformations

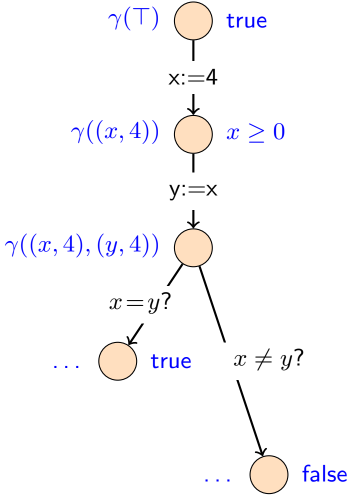


$\{a_1 \dots a_n\}$ solution of $(D^\sharp, T^\sharp)$

A Primer on Certificate Translation

Key Idea

sufficiently strong solution $\leftrightarrow$ preservation along transformations

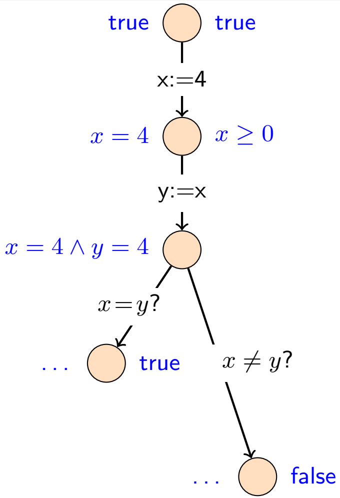


$\{a_1 \dots a_n\}$ solution of $(D^\sharp, T^\sharp)$
$\{\gamma(a_1) \dots \gamma(a_n)\}$ solution of (D, T)

A Primer on Certificate Translation

Key Idea

sufficiently strong solution ↔ preservation along transformations

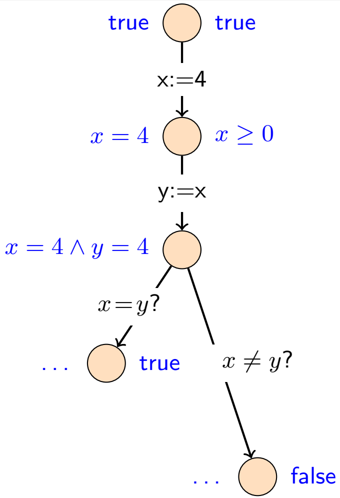


$\{a_1 \dots a_n\}$ solution of $(D^\sharp, T^\sharp)$
$\{\gamma(a_1) \dots \gamma(a_n)\}$ solution of (D, T)

A Primer on Certificate Translation

Key Idea

sufficiently strong solution $\leftrightarrow$ preservation along transformations

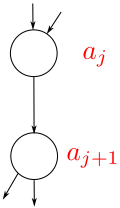


$\{a_1 \ldots a_n\}$ solution of $(D^\sharp, T^\sharp)$
$\{\gamma(a_1) \ldots \gamma(a_n)\}$ solution of (D, T)

$\{a_1 \ldots a_n\}$ solution of (D, T)
$\{b_1 \ldots b_n\}$ solution of (D, T)
$\{a_1 \sqcap b_1 \ldots a_n \sqcap b_n\}$ solution of (D, T)

Certified Setting

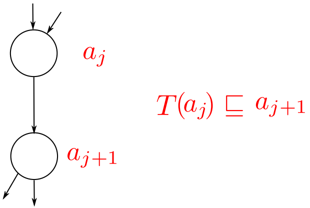
$(a_i)_{i \in \mathcal{N}}$ a solution of (D, T)



$T(a_j) \sqsubseteq a_{j+1}$

Certified Setting

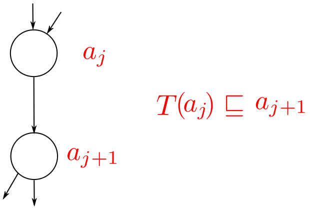
$(a_i)_{i \in \mathcal{N}}$ a solution of (D, T)



- $\sqsubseteq$ is undecidable, e.g. $D =$logic formulae

Certified Setting

$(a_i)_{i\in\mathcal{N}}$ a solution of (D, T)



- $\sqsubseteq$ is undecidable, e.g. $D =$logic formulae
- $\sqsubseteq$ is costly to check.

Proof Algebra

Abstract Certificate Algebra $\mathcal{C}$:

$$\text{if } c \in \mathcal{C}(\vdash a \sqsubseteq a') \text{ then } a \sqsubseteq a'.$$

Proof Algebra

Abstract Certificate Algebra $\mathcal{C}$:

if $c \in \mathcal{C}(\vdash a \sqsubseteq a')$ then $a \sqsubseteq a'$.

$$
\begin{array}{lcl}
\text{axiom} & : & \mathcal{C}(\vdash a \sqsubseteq a) \\
\text{weak}_{\sqcap} & : & \mathcal{C}(\vdash a \sqsubseteq b) \rightarrow \mathcal{C}(\vdash a \sqcap c \sqsubseteq b) \\
\text{weak}_{\sqcup} & : & \mathcal{C}(\vdash a \sqsubseteq b) \rightarrow \mathcal{C}(\vdash a \sqsubseteq b \sqcup c) \\
\text{elim}_{\sqcap} & : & \mathcal{C}(\vdash c \sqcap a \sqsubseteq b) \rightarrow \mathcal{C}(\vdash c \sqsubseteq a) \rightarrow \mathcal{C}(\vdash c \sqsubseteq b) \\
\text{intro}_{\sqcup} & : & \mathcal{C}(\vdash a \sqsubseteq c) \rightarrow \mathcal{C}(\vdash b \sqsubseteq c) \rightarrow \mathcal{C}(\vdash a \sqcup b \sqsubseteq c) \\
\text{intro}_{\sqcap} & : & \mathcal{C}(\vdash a \sqsubseteq b) \rightarrow \mathcal{C}(\vdash a \sqsubseteq c) \rightarrow \mathcal{C}(\vdash a \sqsubseteq b \sqcap c)
\end{array}
$$

Certified Solutions

Definition

$\langle \{a_1 \ldots a_n\}, c\rangle$ is a certified solution if for any edge $\langle i, j\rangle$

$$c(i,j) \in \mathcal{C}(\vdash T_{\langle i,j\rangle}(a_i) \sqsubseteq a_j)$$

Certified Solutions

Definition

$\langle\{a_1 \dots a_n\}, c\rangle$ is a certified solution if for any edge $\langle i, j\rangle$

$$c(i,j) \in \mathcal{C}(\vdash T_{\langle i,j\rangle}(a_i) \sqsubseteq a_j)$$

if $(\{a_1 \dots a_n\}, c_a)$ and $(\{b_1 \dots b_n\}, c_b)$ are certified solutions of D, then $(\{a_1 \sqcap b_1 \dots a_n \sqcap b_n\}, c_a \oplus c_b)$ is a certified solution.

Certified Solutions

Definition

$\langle\{a_1 \dots a_n\}, c\rangle$ is a certified solution if for any edge $\langle i, j\rangle$

$$c(i,j) \in \mathcal{C}(\vdash T_{\langle i,j\rangle}(a_i) \sqsubseteq a_j)$$

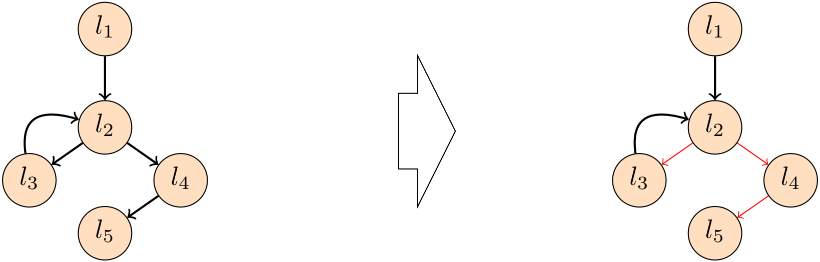
if $(\{a_1 \dots a_n\}, c_a)$ and $(\{b_1 \dots b_n\}, c_b)$ are certified solutions of D, then $(\{a_1 \sqcap b_1 \dots a_n \sqcap b_n\}, c_a \oplus c_b)$ is a certified solution.

if $\{a_1 \dots a_n\}$ is a solution of $(D^\sharp, T^\sharp)$, and cons s.t. for any edge $\langle i, j\rangle$

$$\mathsf{cons}_{\langle i,j\rangle} \in \mathcal{C}(\vdash T_{\langle i,j\rangle}(\gamma(a)) \sqsubseteq \gamma(T^\sharp_{\langle i,j\rangle}(a)))$$

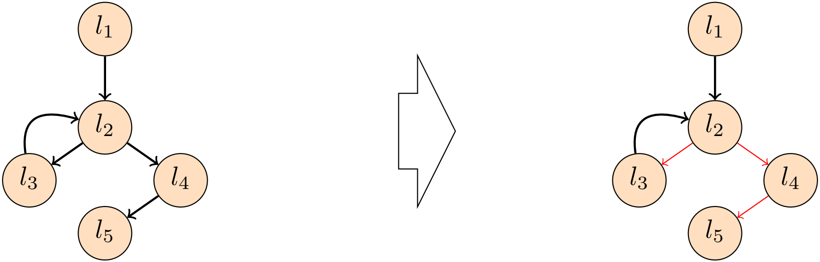
then $(\{\gamma(a_1) \dots \gamma(a_n)\}, c)$ is a certified solution of (D, T) [for some c].

Program Transformation



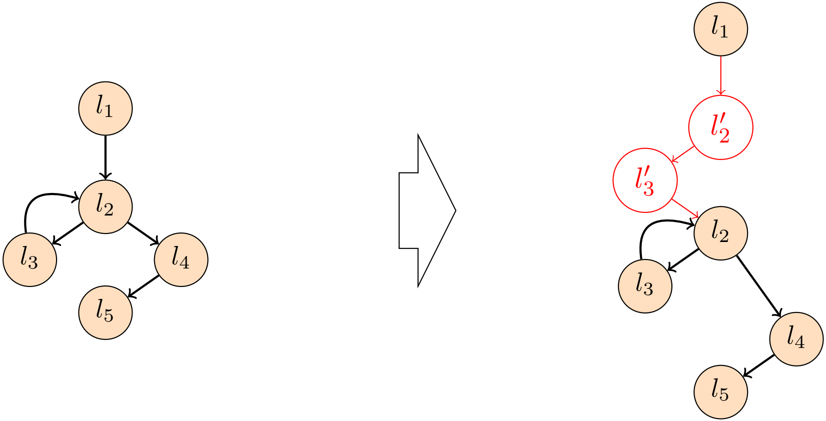
- $T_e \mapsto T'_e$, $e \in \mathcal{E}$
- a proof of $T'_{\langle l_2, l_3 \rangle}(\_) \sqsubseteq a_3 \sqcap T_{\langle l_2, l_3 \rangle}(\_)$

Program Transformation



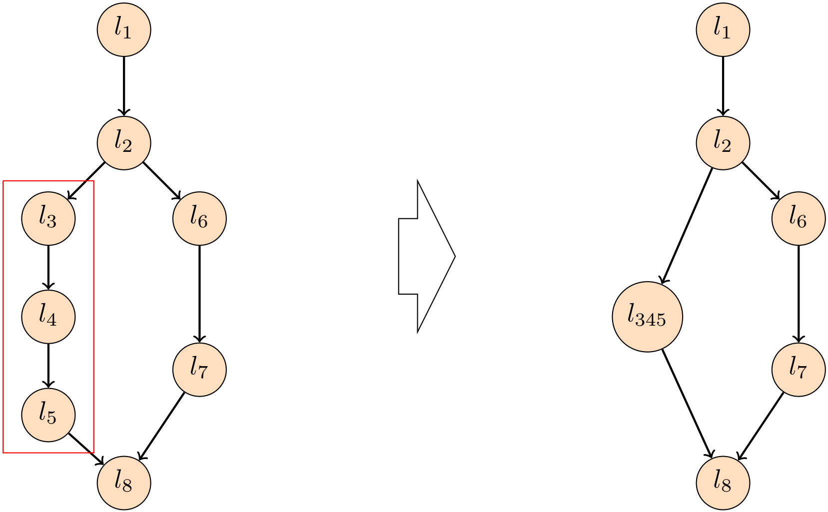
- $T_e \mapsto T'_e$, $e \in \mathcal{E}$
- a proof of $T'_{\langle l_2,l_3\rangle}(\_) \sqsubseteq a_3 \sqcap T_{\langle l_2,l_3\rangle}(\_)$
- const and copy propag / loop induction var strength reduction / common. subexpr elimination / etc.

Code Duplication



- loop unrolling / function inlining

Node Coalescing



In practice, Certificate Translation will

- compute the analysis result that ensures that the transformation is semantics preserving: $S^\sharp$

In practice, Certificate Translation will

- compute the analysis result that ensures that the transformation is semantics preserving: $S^\sharp$
- certify a representation of the analysis: $(\gamma \circ S^\sharp, c_a)$

In practice, Certificate Translation will

- compute the analysis result that ensures that the transformation is semantics preserving: $S^\sharp$
- certify a representation of the analysis: $(\gamma \circ S^\sharp, c_a)$
- certify that $\gamma \circ S^\sharp$ justifies the transformation: justif

In practice, Certificate Translation will

- compute the analysis result that ensures that the transformation is semantics preserving: $S^\sharp$
- certify a representation of the analysis: $(\gamma \circ S^\sharp, c_a)$
- certify that $\gamma \circ S^\sharp$ justifies the transformation: justif
- merges the original certified solution (S, c) with $(\gamma \circ S^\sharp, c_a)$ and justif to generate a certified solution $(S \sqcap \gamma \circ S^\sharp, c \oplus c_a \oplus \text{justif})$

Conclusions

- proposed an abstract model of both program analysis and program verification

Conclusions

- proposed an abstract model of both program analysis and program verification
- extended this model with a notion of certificates

Conclusions

- proposed an abstract model of both program analysis and program verification
- extended this model with a notion of certificates
- studied certifying analyzers and certificate translators in this model

Conclusions

- proposed an abstract model of both program analysis and program verification
- extended this model with a notion of certificates
- studied certifying analyzers and certificate translators in this model
- identify requirements over the analysis and the transformation that can be instantiated to particular frameworks.

Conclusions

- proposed an abstract model of both program analysis and program verification
- extended this model with a notion of certificates
- studied certifying analyzers and certificate translators in this model
- identify requirements over the analysis and the transformation that can be instantiated to particular frameworks.

Thank you.

Example

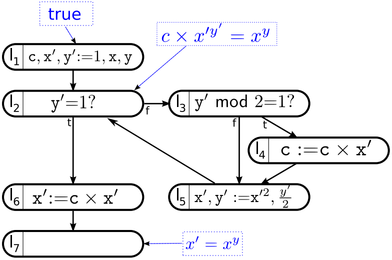


Figure: Annotated program

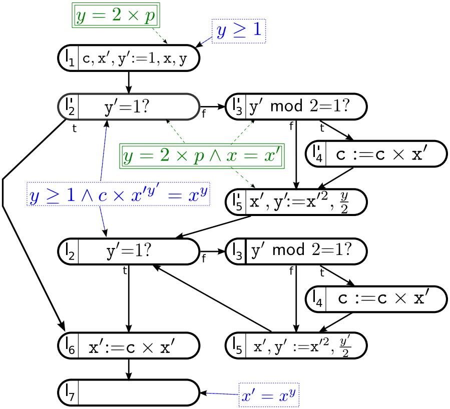


Figure: Program after loop unrolling

Example

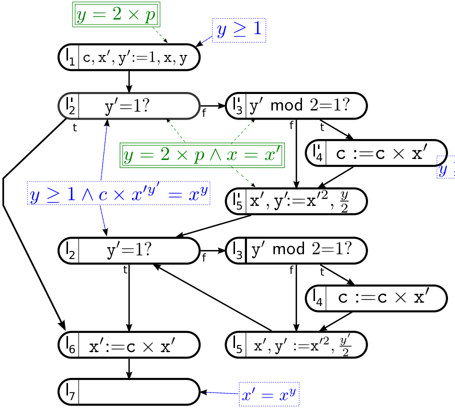


Figure: Program after loop unrolling

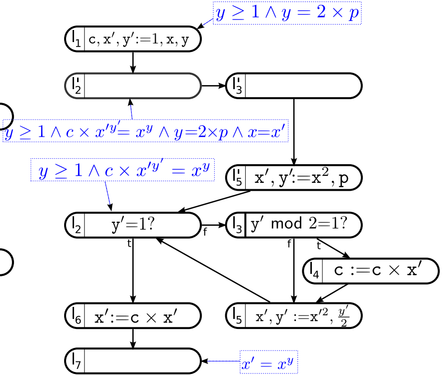


Figure: Program after optimizing transformations

Example

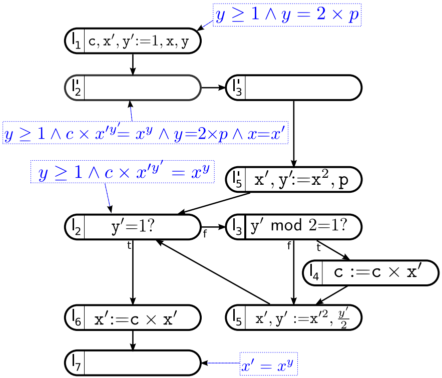


Figure: Program after optimizing transformations

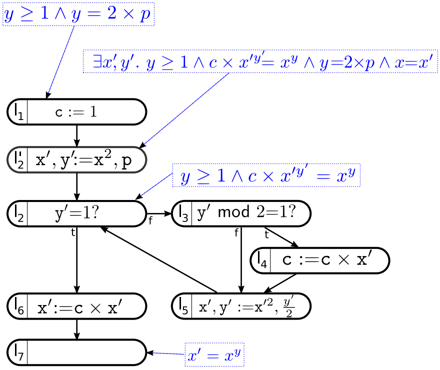


Figure: Node coalescing and dead assignment elimination