Formal Language, Grammar and Set-Constraint-Based Program Analysis by Abstract Interpretation

Patrick COUSOT & Radhia COUSOT

Patrick COUSOT
LIENS – DMI
École Normale Supérieure
75230 Paris cedex 05
France
cousot@dmi.ens.fr

Radhia COUSOT
LIX
CNRS & École Polytechnique
91140 Palaiseau cedex
France
rcousot@lix.polytechnique.fr

Introduction

- There are many kinds of program static analysis methods which are difficult to understand and compare:
  - Data flow analysis,
  - Abstract interpretation,
  - Set based analysis,
  - Type based analysis,
  - Effect systems,
  - etc.
- Our objective is to compare:

  Set Based Analysis

  and **Abstract Interpretation**

What is This Talk About?

- Principle of set-constraint/grammar-based program analysis;
- Principle of abstract interpretation;
- Set-based analysis is an iterative abstract interpretation on a finite abstract domain;
- Beyond set-based analysis: context-sensitive symbolic abstract interpretations.

Set Based Analysis

Principle of Set-Based Program Analysis

- Program analysis:

  static inference of run-time program properties

- Set-based analysis:

$$\begin{array}{ccccc} \text{Program} & & \text{Unsolved} & \textit{resolution} & \text{Solved} \\ & \textit{algorithm} & \text{constraints} & \textit{algorithm} & \text{constraints} \\ P & \longrightarrow & C_P & \longrightarrow & S_P \end{array}$$

INFINITE DOMAIN EXAMPLE[1]

- Program P:

```
X := cons(a, nil);
while X <> nil do
  X := cons(a, X);
```

- Unsolved constraints C_P:

$$[\![\mathtt{X}]\!] \supseteq \mathsf{cons}(\mathsf{a}, \mathsf{nil})$$
$$[\![\mathtt{X}]\!] \supseteq \mathsf{cons}(\mathsf{a}, [\![\mathtt{X}]\!])$$

- Solved constraints S_P:

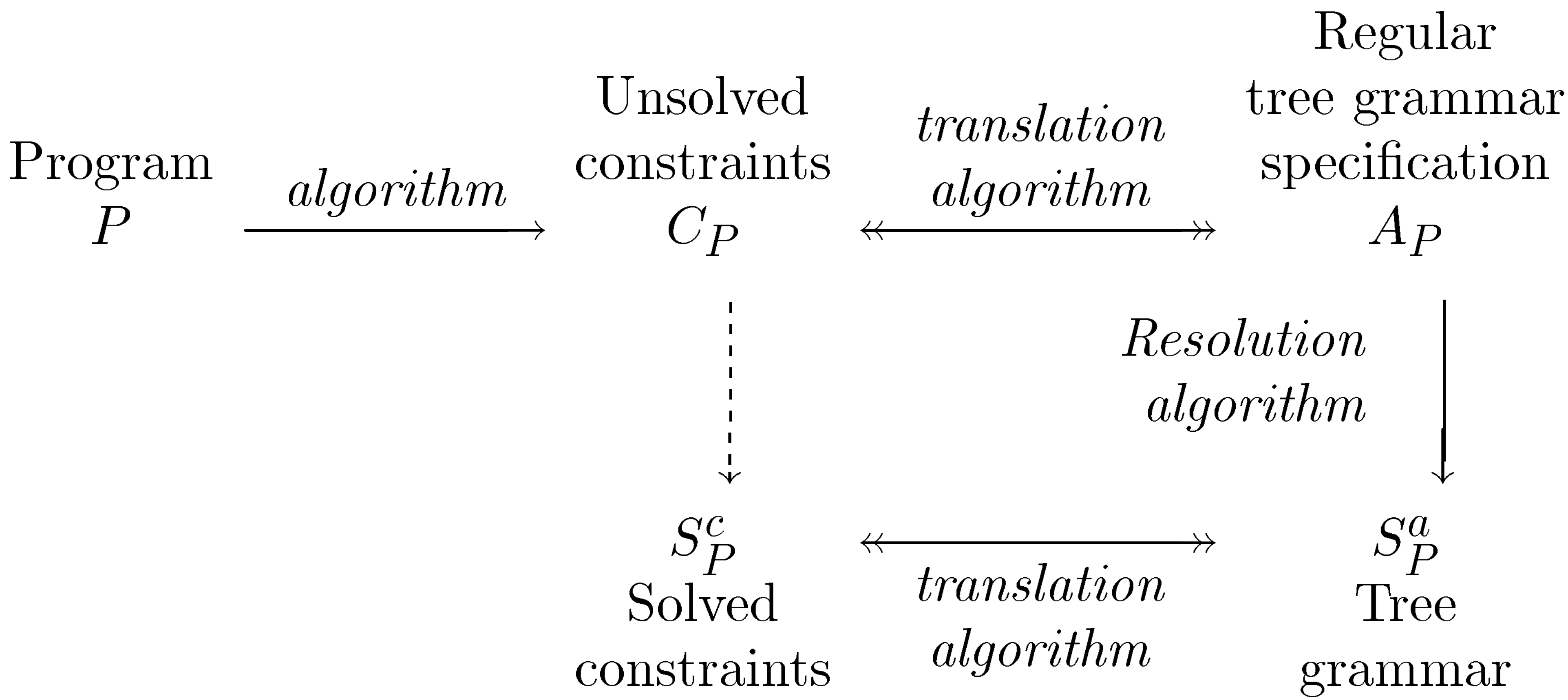
  Already solved, $S_P = C_P$!

[1] Constraint-based program Analysis, A. Aiken & N. Heintze, POPL'95 invited talk.

Implementation of Set-Based Analysis

- Isomorphic to Jones & Muchnick POPL'79 regular tree grammar based analysis;
- Projection → Jones & Muchnick POPL'79 resolution algorithm → polynomial;
- Intersection → auxiliary variables $A_\Delta = \bigcap_{i \in \Delta} X_i$ → exponential resolution algorithm;
- Negation → solutions for a few trivial cases (such as negation of atoms only).

Set Constraints / Regular Tree Grammar



← Specification → ← Implementation →

Abstract Interpretation

Principle of Abstract Interpretation

Program P $\xrightarrow{\textit{algorithm}}$ Abstract semantics specifier $F^{\sharp}_{P}$ $\xrightarrow{\textit{mathematics/ algorithm}}$ Abstract semantics $S^{\sharp}_{P}$

Classical Abstract Interpretation Specification

- Program:

$$P$$

- Abstract semantics specification:

$$\langle D^\sharp_P,\ \sqsubseteq_P,\ \bot_P\rangle, \quad F^\sharp_P \in D^\sharp_P \xrightarrow{\sqsubseteq_P} D^\sharp_P$$

- Abstract semantics:

$$\operatorname{lfp}^{\sqsubseteq_P} F^\sharp_P$$

where:

$$\operatorname{lfp}^{\sqsubseteq_P} F^\sharp_P \stackrel{\text{def}}{=} \bigsqcup_{\lambda} X^\lambda, \quad X^\lambda \stackrel{\text{def}}{=} \bigsqcup_{\eta<\lambda} F^\sharp_P(X^\eta), \quad \bigsqcup\emptyset \stackrel{\text{def}}{=} \bot_P$$

Infinite Domain Example[2]

- Program P:

```
X := cons(a, nil);
while X <> nil do
  X := cons(a, X);
```

- Transformer $F_P^\sharp$:

$$F_P^\sharp(\llbracket X \rrbracket) = \{\texttt{cons(a,nil)}\} \cup \{\texttt{cons(a,}\sigma\texttt{)} \mid \sigma \in \llbracket X \rrbracket\}$$

- Abstract semantics $S_P^\sharp$:

$$S_P^\sharp = \operatorname{lfp}^{\subseteq} F_P^\sharp(\llbracket X \rrbracket) = \{\overbrace{\texttt{[a,}\cdots\texttt{,a]}}^{n \text{ times}} \mid n \geq 1\}$$

[2] Constraint-based program Analysis, A. Aiken & N. Heintze, POPL'95 invited talk.

Infinite Iteration [3]

$[\![X]\!]^0 = \emptyset$

$[\![X]\!]^1 = \{\texttt{[a]}\}$

$[\![X]\!]^2 = \{\texttt{[a]}, \texttt{[a, a]}\}$

$[\![X]\!]^3 = \{\texttt{[a]}, \texttt{[a, a]}, \texttt{[a, a, a]}\}$

and so forth ...

- Remarks [3]:
  - Could insert widening operator around the loop (Yes)
  - But in general this will not yield same result (Yes)

[3] Constraint-based program Analysis, A. Aiken & N. Heintze, POPL'95 invited talk.

Forgot To Say...

– But, one can always design a widening that will lead to an equivalent (or even better) result[4];

Such a widening is provided in the paper for set-constraint/-grammar based analysis;

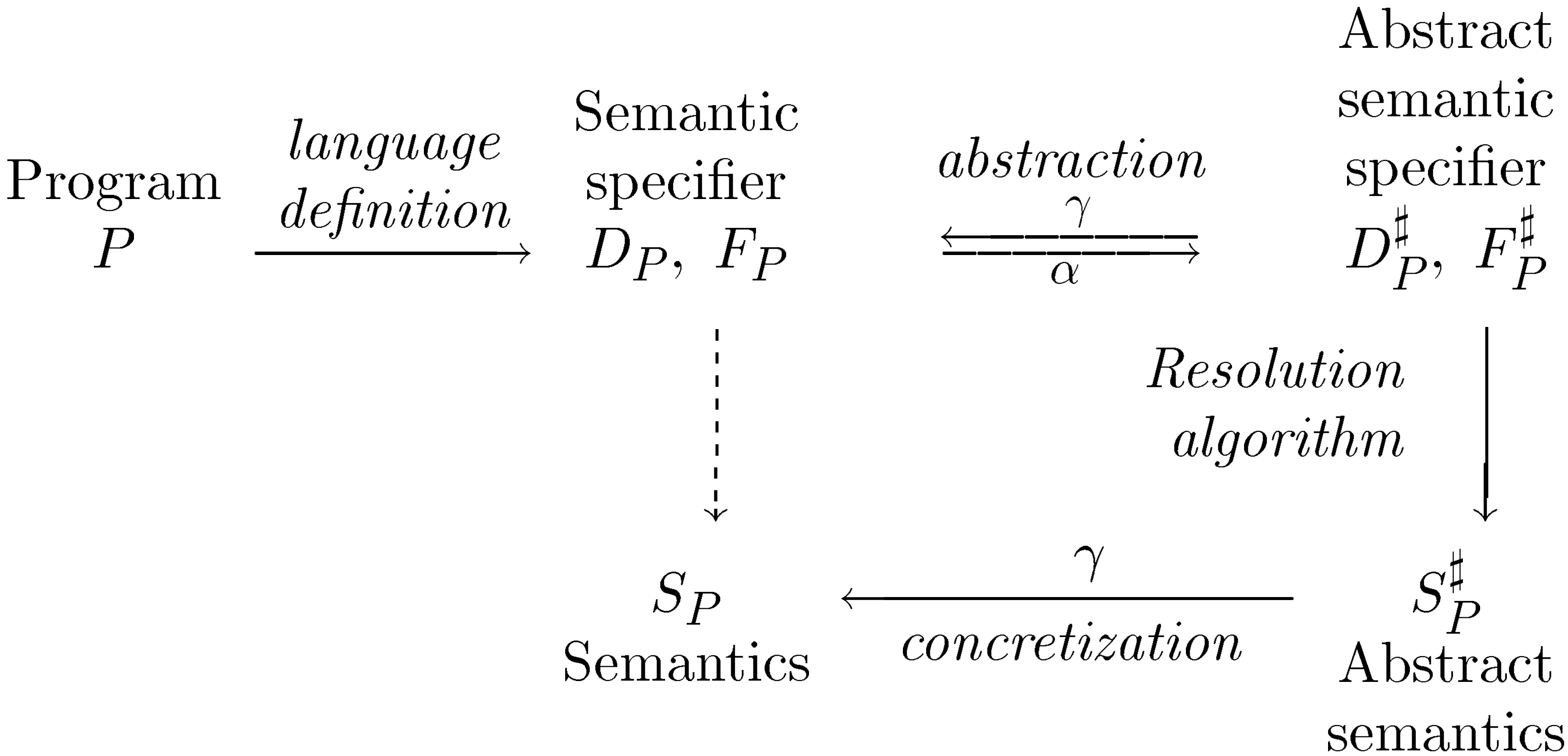
– Moreover, this example is unfair because it compares an abstract interpretation using an *infinite* domain with a set-based analysis using a *finite* abstract domain.

[4] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, invited paper. In M. Bruynooghe and M. Wirsing, editors, *Programming Language Implementation and Logic Programming, Proceedings of the Fourth International Symposium, PLILP'92*, Leuven, (B), 13–17 Aug. 1992, LNCS 631, pages 269–295. Springer-Verlag, 1992.

Objective of the Paper

To show that set based analysis is an abstract interpretation, indeed a trivial one (using an appropriate chaotic least fixpoint iterative computation over a finite domain).

Design of an Abstract Interpretation



- Soundness: $S_P \sqsubseteq_P \gamma(S^\sharp_P)$
- Completeness: $S_P \sqsupseteq_P \gamma(S^\sharp_P)$

The Galois Connection Approach

- Specification of the abstract interpretation:

$$\langle D, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle D^\sharp, \sqsubseteq^\sharp\rangle \qquad \text{Galois connection}$$

$$F^\sharp \overset{\text{def}}{=} \alpha \circ F \circ \gamma$$

$$S^\sharp \overset{\text{def}}{=} \text{lfp}^{\sqsubseteq^\sharp} F^\sharp$$

- Soundness is by construction:

$$S \sqsubseteq \gamma(S^\sharp)$$

- Completeness:

$$\text{if } \alpha \circ F = F^\sharp \circ \alpha \text{ then } S = \gamma(S^\sharp)$$

In Absence of Best Approximation: The Abstraction Function Approach

- Specification of the abstract interpretation:

$$\langle D,\ \sqsubseteq\rangle \xrightarrow{\ \alpha\ } \langle D^\sharp,\ \sqsubseteq^\sharp\rangle \qquad \text{Abstraction function}$$

$$F^\sharp \text{ such that } \alpha \circ F \sqsubseteq^\sharp F^\sharp \circ \alpha$$

$$S^\sharp \stackrel{\text{def}}{=} \operatorname{lfp}^{\sqsubseteq^\sharp} F^\sharp$$

- Soundness is by construction:

$$\alpha(S) \sqsubseteq^\sharp S^\sharp$$

- Completeness:

$$\text{if } \alpha \circ F = F^\sharp \circ \alpha \text{ then } \alpha(S) = S^\sharp$$

Showing that Set Based Analysis <u>is</u> an Abstract Interpretation

$D^\sharp$	$=$	Regular tree grammar in Greibach normal form over a finite vocabulary	$\twoheadleftarrow\!\!\!\longrightarrow\!\!\!\!\twoheadrightarrow$	Set constraints in solved form
$F^\sharp$	$\in$	$D^\sharp \xrightarrow{\subseteq} D^\sharp$, grammar transformer	$\twoheadleftarrow\!\!\!\longrightarrow\!\!\!\!\twoheadrightarrow$	Set constraints in unsolved form
$S^\sharp$	$=$	$\mathrm{lfp}^{\subseteq} F^\sharp$, computed by chaotic iterations	$\twoheadleftarrow\!\!\!\longrightarrow\!\!\!\!\twoheadrightarrow$	Constraint solving algorithm

Set Based Analysis <u>is</u> an Abstract Interpretation

THE FINITE ABSTRACT DOMAIN $D_P^\sharp$

$D_P^\sharp$ is the set of regular tree grammars in Greibach normal form:

$$\begin{cases} \mathcal{X} \rightarrow \mathsf{f}^n(\mathcal{Y}_1, \ldots, \mathcal{Y}_n) \\ \mathcal{X} \rightarrow \mathsf{f}^0 \end{cases}$$

over the finite vocabulary, made of:

- Nonterminals $[\![\texttt{X}]\!]$ where variable, ... `X` appears in program P;
- Finitely many auxiliary nonterminals (intersections, ...);
- Terminals `cons`, `nil`, ... appearing in program P, derived from type declarations in P, ...

Infinitary Abstract Domain ?

- Finite domain $D^\sharp_P$ for each program P

  $\Rightarrow$ this make the analysis feasible

- Infinite domain $D = \bigcup_P D^\sharp_P$ for all programs P

  $\Rightarrow$ this make the analysis impressive

  $\Rightarrow$ but not infinitary!

- Other examples:

  Live variables, constant propagation, ...

CORRESPONDENCE BETWEEN GRAMMARS AND SET CONSTRAINTS

- Grammar: $\boxed{X ::= \mathtt{a}(X) \mid \mathtt{b}}$
- Generated language (Ginsburg & Rice, Schützenberger):

  $\mathcal{X} = \mathrm{lfp}^{\subseteq} F$ where $F(X) = \{\mathtt{a}(\sigma) \mid \sigma \in X\} \cup \{\mathtt{b}\}$
- Fixpoint (Tarski): $\mathrm{lfp}^{\subseteq} F = \bigcap\{X \mid F(X) \subseteq X\}$
- Postfixpoints: $\mathcal{X}$ is the least solution to $[\![\mathtt{X}]\!] \supseteq F([\![\mathtt{X}]\!])$
- Set constraints: $\boxed{[\![\mathtt{X}]\!] \supseteq \mathtt{a}([\![\mathtt{X}]\!]) \cup \mathtt{b}}$

  where: $\mathtt{a}(X) \stackrel{\text{def}}{=} \{\mathtt{a}(\sigma) \mid \sigma \in X\}$

  $\mathtt{b} \stackrel{\text{def}}{=} \{\mathtt{b}\}$

Correspondence Between Unsolved Constraints and Constraint Transformers

(1) Constraint Introduction

Interpret *unsolved constraints* such as:

$$[\![\texttt{X}]\!] \supseteq \mathsf{cons}(\mathsf{a}, [\![\texttt{X}]\!])$$

as *"add this solved constraint"* to the current solved constraints C:

$$F^{\iota}(C) = C \;\cup\; \left\{[\![\texttt{X}]\!] \supseteq \mathsf{cons}(\mathsf{a}, [\![\texttt{X}]\!])\right\}$$

Example

- Program:
```
X := cons(a, nil);
while X <> nil do
  X := cons(a, X);
```

- Unsolved constraints: $[\![\mathtt{X}]\!] \supseteq \mathsf{cons}(\mathsf{a}, \mathsf{nil})$
  $[\![\mathtt{X}]\!] \supseteq \mathsf{cons}(\mathsf{a}, [\![\mathtt{X}]\!])$

  mean:

$$F^{\iota}(C) \;=\; C \;\cup\; \left\{[\![\mathtt{X}]\!] \supseteq \mathsf{cons}(\mathsf{a}, \mathsf{nil})\right\} \;\cup\; \left\{[\![\mathtt{X}]\!] \supseteq \mathsf{cons}(\mathsf{a}, [\![\mathtt{X}]\!])\right\}$$

- Chaotic iteration:

$$X^0 = \emptyset$$
$$X^1 = F^{\iota}(X^0) = \left\{[\![\mathtt{X}]\!] \supseteq \mathsf{cons}(\mathsf{a}, \mathsf{nil})\right\} \;\cup\; \left\{[\![\mathtt{X}]\!] \supseteq \mathsf{cons}(\mathsf{a}, [\![\mathtt{X}]\!])\right\}$$
$$X^1 = F^{\iota}(X^1) = X^2$$

- Equivalent to *"It's solved"*!

Correspondence Between Unsolved Constraints and Constraint Transformers

(2) Standardization

Disjunctive constraints:

$$[\![\mathtt{X}]\!] \supseteq e_1 \cup e_2$$

stands for:

$$F^{\cup}(C) = C \;\cup\; \{[\![\mathtt{X}]\!] \supseteq e_1\} \;\cup\; \{[\![\mathtt{X}]\!] \supseteq e_2\}$$

Correspondence Between <u>Unsolved Constraints</u> and <u>Constraint Transformers</u>

(3) Projection

A projection:

$$[\![\mathtt{X}]\!] \supseteq \mathsf{cons}^{-1}([\![\mathtt{Y}]\!])$$

stands for:

$$F^{-1}(C) = C \;\cup\; \left\{[\![\mathtt{X}]\!] \supseteq e_1 \;\middle|\; [\![\mathtt{Y}]\!] \supseteq \mathtt{cons}(e_1, e_2) \in C\right\}$$

Chaotic Iteration Isomorphic to Constraint Solving Algorithm

Solve:

$$C = C \cup F^{\iota}(C) \cup F^{\cup}(C) \cup F^{-1}(C)$$

with following chaotic iteration:

$C := F^{\iota}(\emptyset)$;	introduce solved constraints
$C := F^{\cup}(C)$;	standardize
Iterate	solve projections
$\quad C := F^{-1}(C)$	
Until stabilization;	

Beyond Set-based Analysis

Combination of Symbolic and Numeric Constraints

$$\begin{cases} [\![\mathtt{X}]\!] \supseteq \mathsf{cons}(n, \mathsf{nil}) & \text{symbolic constraints} \\ [\![\mathtt{X}]\!] \supseteq \mathsf{cons}(m, [\![\mathtt{X}]\!]) & \\ m \geq n + 1 & \text{numerical constraints} \\ m = n \bmod 2 & \end{cases}$$

- m, n are pseudo-terminals in the grammar;
- Numerical constraints universally quantified over all instances of the pseudo-terminals.

Context Sensitive Constraints

$$\begin{cases} X \xrightarrow{n} \mathsf{f}(X, Y, \ldots) & \text{grammar rules with counters} \\ Y \xrightarrow{m} \mathsf{g}(X, Y, \ldots) & \\ \alpha n + \beta m = \gamma & \text{numerical constraints on counters} \end{cases}$$

- Count number of uses of each grammar rule in derivations;
- Linear equality constraints on these counters;
- Can express context-sensitive constraints (e.g. lists have equal length);
- Infinite abstract domain satisfying the ascending chain condition.

Example of Context Sensitive Analysis

- ```
  f(N) = if (N <= 0) then
           cons(0, cons(0, cons(0, nil)))
         else let X = f(N-1) in
           cons(a(hd(X)), cons(b(hd(tl(X))),
              cons(c(hd(tl(tl(X)))), nil)));
  ```
- ```
  X := cons(0, cons(0, cons(0, nil)));
  while true do
    X := cons(a(hd(X)), cons(b(hd(tl(X))),
            cons(c(hd(tl(tl(X)))), nil)));
  od;
  ```
- Set based analysis: $\mathtt{a}^\star\mathtt{b}^\star\mathtt{c}^\star$
- Context sensitive analysis: $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 1\}$

Conclusion: Interest of Understanding Set-Based Analysis as an Abstract Interpretation

- Better understanding of set-based analysis;
- Systematic design method using an abstraction function;
- Combinations with other abstract domains;
- Context-sensitive analyses which expressive power is far beyond set-based analysis.