Constraint-based Invariant Inference

Ravichandhran Madhavan,

Viktor Kuncak, EPFL, Switzerland

Invariants

- **Dictionary Meaning:** A function, quantity, or **property** which remains unchanged
- Property (in our context): a predicate that holds for some, all, or no states
- Invariant is a property of a program
 - at a specific program location
 - that holds for every program state that reaches the program point
- Specifications are invariants at exit points of programs or procedures
- Also called reachability properties.

Invariants

x = 0y = nwhile (y > 0) { x = x + 1y = y - 1} //invariant: x+y = n//invariant: $y \ge 0 \implies x \le n$

Inductive Invariants

$$x = 0$$

$$y = n$$

$$//x+y = n$$
while (y > 0) {
 //x+y = n \land y > 0
 x = x + 1
 //x+y = n+1
 y = y - 1
 y = y - 1
 //x+y = n
}

//invariant: x+y = n

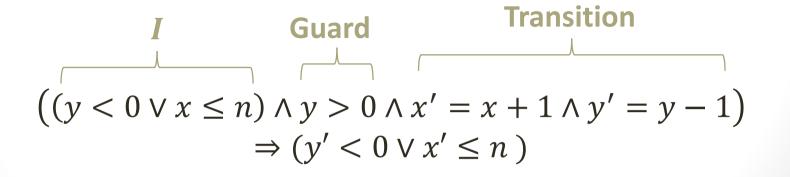
Not all Invariants are Inductive

Inductive Strengthening

Formulating Inductiveness

Generally referred to as the verification condition (VC)

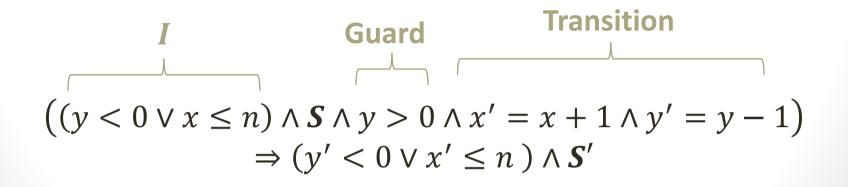
$$(x = 0 \land y = n) \Rightarrow (y < 0 \lor x \le n)$$



Formulating Inductive Strengthening x = 0 y = n while(y > 0){ x = x + 1

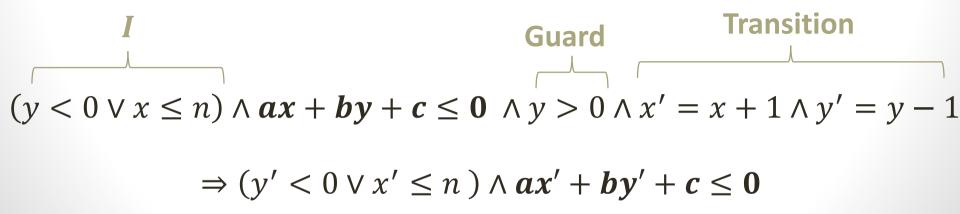
y = y - 1
} //invariant: y>=0 => x<=n</pre>

$$(x = 0 \land y = n) \Rightarrow (y < 0 \lor x \le n) \land S$$



Finding Linear Invariants [Colon et al. CAV '03]

$$(x = 0 \land y = n) \Rightarrow (y < 0 \lor x \le n) \land ax + by + c \le 0$$



Finding Template Coefficients

Farkas' Lemma: A conjunction of linear inequalities is unsatisfiable iff we can derive **1** <= **0** by performing the following operations:

- Multiplying the inequalities by a non-negative constant
- Adding two inequalities
- Adding (or subtracting) a non-negative constant to one side

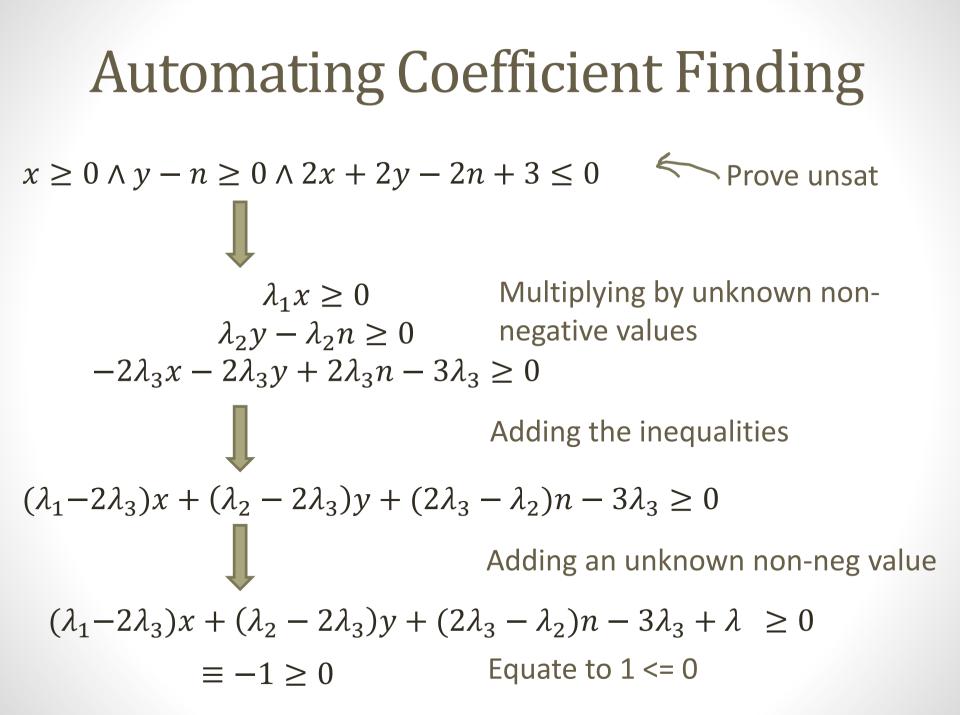
Farkas' Lemma Example $x \ge 0 \land y \ge n \land 2x + 2y - 2n + 3 \le 0$

 $x + 0y + 0n + 0 \ge 0$ $0x + y - n + 0 \ge 0$ $-2x - 2y + 2n - 3 \ge 0$ Multiply first and second equations by 2, Add 2 to RHS of last equation and add them

 $-1 \ge 0$

Farkas' Lemma: A conjunction of linear inequalities (over reals) is unsatisfiable iff we can derive **1** <= **0** by performing the following operations:

- Multiplying the inequalities by a non-negative constant
- Adding two inequalities
- Adding (or subtracting) a non-negative constant to one side



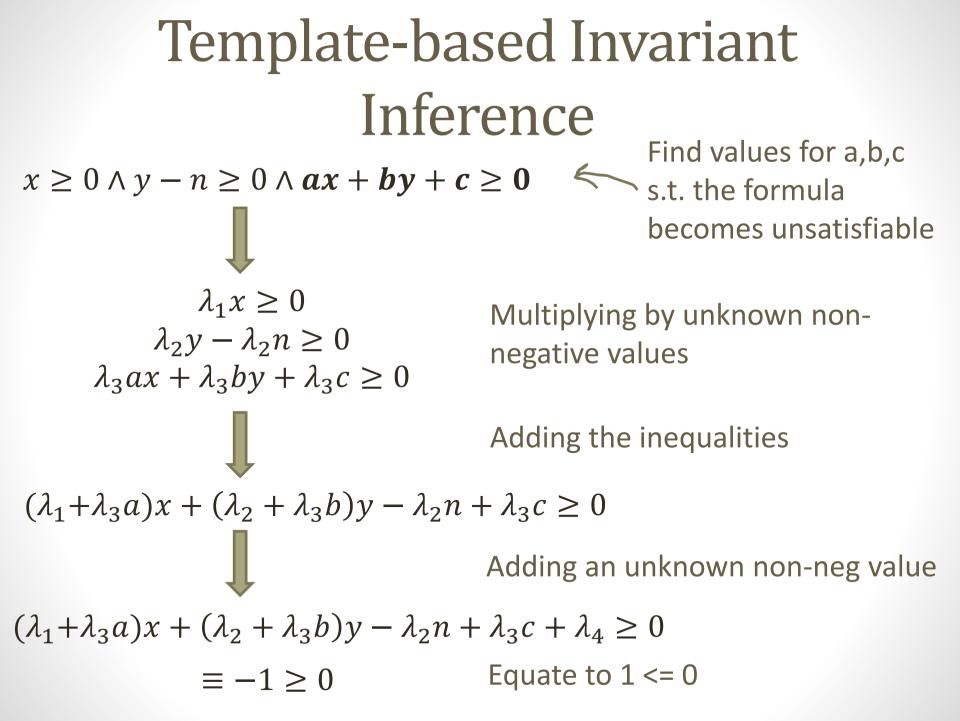
Automating Coefficient Finding [Cont.]

 $(\lambda_1 - 2\lambda_3)x + (\lambda_2 - 2\lambda_3)y + (2\lambda_3 - \lambda_2)n - 3\lambda_3 + \lambda \ge 0$ $\equiv -1 > 0$

 $\lambda_{1} - 2\lambda_{3} = 0$ $\lambda_{2} - 2\lambda_{3} = 0$ $2\lambda_{3} - \lambda_{2} = 0$ $-3\lambda_{3} + \lambda = -1$

Every solution for the constraints will make the inequalities unsatisfiable

$$\lambda_1=2$$
 , $\lambda_2=2$, $\lambda_3=1, \lambda=2$



Farkas' Constraints [Cont.]

 $(\lambda_1 + \lambda_3 a)x + (\lambda_2 + \lambda_3 b)y - \lambda_2 n + \lambda_3 c + \lambda_4 \ge 0$ $\equiv -1 \ge 0$

 $\lambda_{1} + \lambda_{3}a = 0$ $\lambda_{2} + \lambda_{3}b = 0$ $-\lambda_{2} = 0$ $\lambda_{3}c + \lambda_{4} = -1$

Every solution for the constraints will make the inequalities unsatisfiable

$$b = 0, a = -1, c = -1,$$

 $\lambda_1 = 1, \lambda_2 = 0,$
 $\lambda_3 = 1, \lambda_4 = 0$

In summary

- We had a formula of the form: $A[\mathbf{x}] \wedge B[\mathbf{a}, \mathbf{x}] \Rightarrow C[\mathbf{a}, \mathbf{x}]$
- We wanted to find a value for *a* that will make the implication hold for all x
- In other words, we are trying to find a satisfiable assignment for a quantified formula.
- Farkas' Lemma converts it to satisfiability of quantifier-free non-linear real constraints

Limitations

The Farkas' Lemma approach provides a way to find linear invariants for programs that

- do not have many disjunctions
- do not have functions
- do not have data structures
- do not have nonlinear arithmetic

Further Reading and Software

We developed an approach that addresses some of these limitations. For more details see:

"Symbolic Resource Bounds Inference For Functional Programs", CAV 2014: <u>pdf</u> , <u>slides</u>

An extension of Leon (a slightly old version) that supports templates: Orb : <u>http://lara.epfl.ch/w/rbound</u>

- More Related Works
 - "Linear invariant generation using non-linear constraint solving.", Colon et al., CAV 2003
 - "Program analysis as constraint solving.", S. Gulwani et al., PLDI 2008
 - "Constraint solving for interpolation.", A.Rybalchenko et al., VMCAI 2007
 - "Non-linear loop invariant generation using grobner bases."
 Sankaranarayanan et al., POPL 2004