SMT: Linear Arithmetic and Combination of Theories

ALC

IST, ULisboa

Outline

Outline

Theories: Real / Integer Linear Arithmetic

Theories Solver for RLA: The Simplex Method

Theories Solver for RLA: Fourier-Motzkin

Theories: Integer Linear Arithmetic

SMT: Recap

Theory Combinations

Context – SMT Lazy Solving

input : formula φ in theory $\mathcal{T}$
output: truth value

$\alpha \leftarrow \mathcal{T}2\mathcal{B}(\varphi)$ // abstract input formula
while true **do**
    $(\text{res}, \tau) \leftarrow \text{SAT}(\alpha)$ // Boolean model
    if res $=$ false **then return** false
    $\mathcal{L} \leftarrow \bigcup_{l \in \tau} \mathcal{B}2\mathcal{T}(l)$ // convert to theory model
    $(\text{res}, \mathcal{L}') \leftarrow \mathcal{T}\text{-SAT}(\mathcal{L})$ // theory check
    if res $=$ true **then return** true
    $\alpha \leftarrow \alpha \wedge \bigvee_{l \in \mathcal{L}'} \neg \mathcal{T}2\mathcal{B}(l)$ // block explanation
end

Linear arithmetic

$$\begin{array}{ll} \text{atom}: & \text{sum op sum} \\ \text{op}: & \leq \mid < \mid = \\ \text{sum}: & \text{term} + \text{sum} \\ \text{term}: & \text{uninterpretedConstant} \mid \text{interpretedConstant} \end{array}$$

- Integers: **ILA**
- Reals: **RLA**

A simple example

- Example RLA/ILA formula:

$$
\begin{array}{l}
((x_1 + x_2 \geq 2 \vee x_3 + x_4 \geq 3)\wedge \\
((x_1 + x_2 \leq 1 \vee x_3 + x_4 \leq 2)\wedge \\
((x_5 + x_6 \geq 2 \vee x_7 + x_8 \geq 3)\wedge \\
((x_5 + x_6 \leq 1 \vee x_7 + x_8 \leq 2)
\end{array}
$$

- Represent Boolean structure as CNF formula:

$$(a \vee b) \wedge (c \vee d) \wedge (e \vee f) \wedge (g \vee h)$$

- Interaction between SAT solver & theory solver (RLA/ILA):

SAT Outcome	Model/Core	(R | I)LA Outcome	Explanation clause (sent to SAT solver)

A simple example

- Example RLA/ILA formula:

$$
\begin{array}{l}
((x_1 + x_2 \geq 2 \vee x_3 + x_4 \geq 3)\wedge \\
((x_1 + x_2 \leq 1 \vee x_3 + x_4 \leq 2)\wedge \\
((x_5 + x_6 \geq 2 \vee x_7 + x_8 \geq 3)\wedge \\
((x_5 + x_6 \leq 1 \vee x_7 + x_8 \leq 2)
\end{array}
$$

- Represent Boolean structure as CNF formula:

$$(a \vee b) \wedge (c \vee d) \wedge (e \vee f) \wedge (g \vee h)$$

- Interaction between SAT solver & theory solver (RLA/ILA):

SAT Outcome	Model/Core	(R | I)LA Outcome	Explanation clause (sent to SAT solver)
SAT	$\{a, \bar{b}, c, \bar{d}, e, \bar{f}, g, \bar{h}\}$	UNSAT	$(\bar{a} \vee \bar{c})$

A simple example

- Example RLA/ILA formula:

$$((x_1 + x_2 \geq 2 \vee x_3 + x_4 \geq 3)\wedge \\ ((x_1 + x_2 \leq 1 \vee x_3 + x_4 \leq 2)\wedge \\ ((x_5 + x_6 \geq 2 \vee x_7 + x_8 \geq 3)\wedge \\ ((x_5 + x_6 \leq 1 \vee x_7 + x_8 \leq 2)$$

- Represent Boolean structure as CNF formula:

$$(a \vee b) \wedge (c \vee d) \wedge (e \vee f) \wedge (g \vee h)$$

- Interaction between SAT solver & theory solver (RLA/ILA):

SAT Outcome	Model/Core	(R | I)LA Outcome	Explanation clause (sent to SAT solver)
SAT	$\{a, \bar{b}, c, \bar{d}, e, \bar{f}, g, \bar{h}\}$	UNSAT	$(\bar{a} \vee \bar{c})$
SAT	$\{a, \bar{b}, \bar{c}, d, e, \bar{f}, g, \bar{h}\}$	UNSAT	$(\bar{e} \vee \bar{g})$

A simple example

- Example RLA/ILA formula:

$$\begin{array}{l} ((x_1 + x_2 \geq 2 \vee x_3 + x_4 \geq 3) \wedge \\ ((x_1 + x_2 \leq 1 \vee x_3 + x_4 \leq 2) \wedge \\ ((x_5 + x_6 \geq 2 \vee x_7 + x_8 \geq 3) \wedge \\ ((x_5 + x_6 \leq 1 \vee x_7 + x_8 \leq 2) \end{array}$$

- Represent Boolean structure as CNF formula:

$$(a \vee b) \wedge (c \vee d) \wedge (e \vee f) \wedge (g \vee h)$$

- Interaction between SAT solver & theory solver (RLA/ILA):

SAT Outcome	Model/Core	(R | I)LA Outcome	Explanation clause (sent to SAT solver)
SAT	$\{a, \bar{b}, c, \bar{d}, e, \bar{f}, g, \bar{h}\}$	UNSAT	$(\bar{a} \vee \bar{c})$
SAT	$\{a, \bar{b}, \bar{c}, d, e, \bar{f}, g, \bar{h}\}$	UNSAT	$(\bar{e} \vee \bar{g})$
SAT	$\{a, \bar{b}, \bar{c}, d, e, \bar{f}, \bar{g}, h\}$	SAT	Done

Outline

Using Simplex

Recap. Ch. 29 of Introduction to Algorithms (Cormen et al. – 3^{rd} Ed.)

$$\begin{array}{rcrcrcr} x_1 & + & x_2 & & & \leq & 0 \\ x_1 & & & + & x_3 & \leq & 0 \\ & & & - & x_3 & \leq & -1 \\ & & x_1, x_2, x_3 & & & \geq & 0 \end{array}$$

Using Simplex

Recap. Ch. 29 of Introduction to Algorithms (Cormen et al. – 3^{rd} Ed.)

$$\begin{array}{rcrcrcr} x_1 & + & x_2 & & & \leq & 0 \\ x_1 & & & + & x_3 & \leq & 0 \\ & & & - & x_3 & \leq & -1 \\ & & x_1, x_2, x_3 & & & \geq & 0 \end{array}$$

Basic solution does not work. Build auxiliary linear program:

maximize $-x_0$

such that

$$\begin{array}{rcrcrcrcr} x_1 & + & x_2 & & & - & x_0 & \leq & 0 \\ x_1 & & & + & x_3 & - & x_0 & \leq & 0 \\ & & & - & x_3 & - & x_0 & \leq & -1 \\ & & & & x_1, x_2, x_3, x_0 & & & \geq & 0 \end{array}$$

Using Simplex

Slack form:

$$
\begin{array}{rcrcrcrcrcr}
z & = & & & & & & & & - & x_0 \\
s_1 & = & 0 & - & x_1 & - & x_2 & & & + & x_0 \\
s_2 & = & 0 & - & x_1 & & & - & x_3 & + & x_0 \\
s_3 & = & -1 & & & & & + & x_3 & + & x_0
\end{array}
$$

Pivot between x_0 and s_3

$$
\begin{array}{rcrcrcrcrcr}
z & = & -1 & & & & & + & x_3 & - & s_3 \\
s_1 & = & 1 & - & x_1 & - & x_2 & - & x_3 & + & s_3 \\
s_2 & = & 1 & - & x_1 & & & - & 2x_3 & + & s_3 \\
x_0 & = & 1 & & & & & - & x_3 & + & s_3
\end{array}
$$

Using Simplex

Slack form:

$$
\begin{array}{rcrcrcrcrcr}
z & = & -1 & & & & & + & x_3 & - & s_3 \\
s_1 & = & 1 & - & x_1 & - & x_2 & - & x_3 & + & s_3 \\
s_2 & = & 1 & - & x_1 & & & - & 2x_3 & + & s_3 \\
x_0 & = & 1 & & & & & - & x_3 & + & s_3
\end{array}
$$

Pivot between x_3 and s_2

$$
\begin{array}{rcrcrcrcrcr}
z & = & -\frac{1}{2} & - & \frac{x_1}{2} & & & - & \frac{s_2}{2} & - & \frac{s_3}{2} \\
s_1 & = & \frac{1}{2} & - & \frac{x_1}{2} & - & x_2 & + & \frac{s_2}{2} & + & \frac{s_3}{2} \\
x_3 & = & \frac{1}{2} & - & \frac{x_1}{2} & & & - & \frac{s_2}{2} & + & \frac{s_3}{2} \\
x_0 & = & \frac{1}{2} & + & \frac{x_1}{2} & & & + & \frac{s_2}{2} & + & \frac{s_3}{2}
\end{array}
$$

Using Simplex

- Optimal solution was found with $x_0 \neq 0$. RLA formula is unsatisfiable
  - If $x_0 = 0$, then formula would be satisfiable
- Unsatisfiable core can be determined by analysing final slack form
  - Variable s_i is a non-basic variable (right-hand side) in final slack form means constraint i belongs to the unsatisfiable core
- In our example, the unsatisfiable core would be:

$$\begin{array}{rcrcr} x_1 & + & x_3 & \leq & 0 \\ & - & x_3 & \leq & -1 \end{array}$$

Remarks on Simplex

Standard Simplex only supports weak inequalities ($\leq$).
But we also have $=, <$.

- Rewrite equalities with two inequalities
- Rewrite strict inequalities as weak with additional variable:
  1. To decide: $\bigwedge_i \alpha_{i1}x_1 + \cdots + \alpha_{in}x_n < \beta_i$
  2. Maximize x_{n+1} subject to $\bigwedge_i \alpha_{i1}x_1 + \cdots + \alpha_{in}x_n + x_{n+1} \leq \beta_i$
  3. If optimum positive, original satisfiable.

Outline

Using Fourier-Motzkin (FM)

- Alternative to Simplex
- Idea:
  - If $A \leq x$ and $x \leq B$ then $A \leq B$
  - Elimination by enforcing all bounds to be nonempty:

$$\left(\bigwedge_i A_i \leq x \wedge \bigwedge_j x \leq B_j \right) \Leftrightarrow \bigwedge_{i,j} A_i \leq B_j$$

- Complexity: Elimination of one variable is quadratic. Elimination of n variables: $m^2/4, m^4/16, \ldots m^{2^n}/4^n$

FM phase 1 eliminate equalities

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i$$

- Pick equality i and remove x_n

FM phase 1 eliminate equalities

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i$$

$$x_n = \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} \cdot x_j$$

- Pick equality i and remove x_n

FM phase 1 eliminate equalities

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i$$

$$x_n = \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} \cdot x_j$$

$$\bigwedge_{i=1}^{m} \sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i$$

- Pick equality i and remove x_n

FM phase 2: Variable Elimination

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i$$

- Pick inequality i and remove x_n

FM phase 2: Variable Elimination

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i$$

$$a_{in} \cdot x_n \leq b_i - \sum_{j=1}^{n-1} a_{ij} \cdot x_j$$

- Pick inequality i and remove x_n

FM phase 2: Variable Elimination

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i$$

$$a_{in} \cdot x_n \leq b_i - \sum_{j=1}^{n-1} a_{ij} \cdot x_j$$

$$x_n \leq \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} \cdot x_j$$

- Pick inequality i and remove x_n

FM phase 2: Variable Elimination

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i$$

$$a_{in} \cdot x_n \leq b_i - \sum_{j=1}^{n-1} a_{ij} \cdot x_j$$

$$x_n \leq \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} \cdot x_j$$

$$\beta_l \leq x_n \leq \beta_u$$

- Pick inequality i and remove x_n
- Combine **all** pairs l and u

FM phase 2: Variable Elimination

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i$$

$$a_{in} \cdot x_n \leq b_i - \sum_{j=1}^{n-1} a_{ij} \cdot x_j$$

$$x_n \leq \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} \cdot x_j$$

$$\beta_l \leq x_n \leq \beta_u$$

$$\beta_l \leq \beta_u$$

- Pick inequality i and remove x_n
- Combine **all** pairs l and u
- If variable unbounded, then **remove** variable

FM: An example

- Initial formula:

$$\begin{array}{rcrcrl} x_1 & - & x_2 & & & \leq \quad 0 \\ x_1 & & & - & x_3 & \leq \quad 0 \\ -x_1 & + & x_2 & + & 2x_3 & \leq \quad 0 \\ & & & & -x_3 & \leq \quad -1 \end{array}$$

- Eliminate x_1:

FM: An example

- Initial formula:

$$\begin{array}{rcrcrcr} x_1 & - & x_2 & & & \leq & 0 \\ x_1 & & & - & x_3 & \leq & 0 \\ -x_1 & + & x_2 & + & 2x_3 & \leq & 0 \\ & & & & -x_3 & \leq & -1 \end{array}$$

- Eliminate x_1:

Bounds: $(x_2 + 2x_3) \leq x_1 \leq \min(x_2, x_3)$

$$\begin{array}{rcrcr} & & 2x_3 & \leq & 0 \\ x_2 & + & x_3 & \leq & 0 \\ & & -x_3 & \leq & -1 \end{array}$$

FM contd.

- Eliminate x_2 (unbounded removed):

$$\begin{array}{rcr} 2x_3 & \leq & 0 \\ -x_3 & \leq & -1 \end{array}$$

- Eliminate x_3:

Bounds: $1 \leq x_3 \leq 0$

$$1 \leq 0$$

FM contd.

- Eliminate x_2 (unbounded removed):

$$\begin{array}{rcr} 2x_3 & \leq & 0 \\ -x_3 & \leq & -1 \end{array}$$

- Eliminate x_3:

Bounds: $1 \leq x_3 \leq 0$

$$1 \leq 0$$

- Formula is unsatisfiable
- Unsatisfiable core can be determined by a trace back on the dependencies of the final constraints

Outline

Decision procedures for integer domains

- Number of techniques, integer linear programming (ILP) solvers
- Several techniques based on
  - solving in reals (relaxation)
  - blocking non-integer solutions

Branch and Bound in Integer Linear Programming

- Main steps:

Branch and Bound in Integer Linear Programming

- Main steps:
  1. Solve relaxed linear program (e.g. Simplex)

Branch and Bound in Integer Linear Programming

- Main steps:
  1. Solve relaxed linear program (e.g. Simplex)
  2. Pick real valued variable $x_i = v_i$ and create two new constraints:
     - $x_i \leq \lfloor v_i \rfloor$
     - $x_i \geq \lceil v_i \rceil$
  3. Split on the two constraints (one and only one must hold)

Outline

SMT: Approaches

- Two main approaches: eager and lazy
- Eager techniques convert to a single SAT call
- Lazy techniques call SAT repeatedly

SMT: Eager

- Small model property: in some theories, if there is a model there is a small one (bounded)
- Example: Satisfiable formulas in *Integer difference logic*'s must have a bounded solution
- Ackermann reduction: encode congruence axioms as implications

SMT: Lazy

- SAT handles the propositional structure
- Theory solver checks models from the SAT solver within the theory
- Example: *equivalence with uninterpreted functions* solved by congruence closure
- Nelson-Oppen closure enables combining multiple theory solvers

Outline

Context

Language

- Signature $\Sigma = \Sigma_F \cup \Sigma_P$

Context

Language

- Signature $\Sigma = \Sigma_F \cup \Sigma_P$
- Every predicate $p \in \Sigma_P$ and function $f \in \Sigma_F$ has a fixed arity.

Context

Language

- Signature $\Sigma = \Sigma_F \cup \Sigma_P$
- Every predicate $p \in \Sigma_P$ and function $f \in \Sigma_F$ has a fixed arity.
- Functions with arity 0 are called constants.

Context

Language

- Signature $\Sigma = \Sigma_F \cup \Sigma_P$
- Every predicate $p \in \Sigma_P$ and function $f \in \Sigma_F$ has a fixed arity.
- Functions with arity 0 are called constants.
- Countable set of variables $\mathcal{V}$, disjoint from Σ.

Context

Language

- Signature $\Sigma = \Sigma_F \cup \Sigma_P$
- Every predicate $p \in \Sigma_P$ and function $f \in \Sigma_F$ has a fixed arity.
- Functions with arity 0 are called constants.
- Countable set of variables $\mathcal{V}$, disjoint from Σ.

Semantics

Context

Language

- Signature $\Sigma = \Sigma_F \cup \Sigma_P$
- Every predicate $p \in \Sigma_P$ and function $f \in \Sigma_F$ has a fixed arity.
- Functions with arity 0 are called constants.
- Countable set of variables $\mathcal{V}$, disjoint from Σ.

Semantics

- Σ-model (or structure): $\mathcal{A} = (A, \mathcal{I})$

Context

Language

- Signature $\Sigma = \Sigma_F \cup \Sigma_P$
- Every predicate $p \in \Sigma_P$ and function $f \in \Sigma_F$ has a fixed arity.
- Functions with arity 0 are called constants.
- Countable set of variables $\mathcal{V}$, disjoint from Σ.

Semantics

- Σ-model (or structure): $\mathcal{A} = (A, \mathcal{I})$
- A is the universe (or domain)

Context

Language

- Signature $\Sigma = \Sigma_F \cup \Sigma_P$
- Every predicate $p \in \Sigma_P$ and function $f \in \Sigma_F$ has a fixed arity.
- Functions with arity 0 are called constants.
- Countable set of variables $\mathcal{V}$, disjoint from Σ.

Semantics

- Σ-model (or structure): $\mathcal{A} = (A, \mathcal{I})$
- A is the universe (or domain)
- $\mathcal{I}$ is the interpretation

Context – Models and Satisfiability

- A model $\mathcal{A}$ satisfies ϕ iff ϕ evaluates to true under standard semantics

Context – Models and Satisfiability

- A model $\mathcal{A}$ satisfies ϕ iff ϕ evaluates to true under standard semantics
- For convenience, free variables are treated as uninterpreted constants.

Context – Models and Satisfiability

- A model $\mathcal{A}$ satisfies ϕ iff ϕ evaluates to true under standard semantics
- For convenience, free variables are treated as uninterpreted constants.
- In SMT, for a given theory $\mathcal{T}$, a formula ϕ is $\mathcal{T}$-satisfied by a model $\mathcal{A}$ if $\mathcal{A}$ satisfies ϕ and it also satisfies $\mathcal{T}$.

Context – Models and Satisfiability

- A model $\mathcal{A}$ satisfies ϕ iff ϕ evaluates to true under standard semantics
- For convenience, free variables are treated as uninterpreted constants.
- In SMT, for a given theory $\mathcal{T}$, a formula ϕ is $\mathcal{T}$-satisfied by a model $\mathcal{A}$ if $\mathcal{A}$ satisfies ϕ and it also satisfies $\mathcal{T}$.
- A formula is $\mathcal{T}$-satisfiable if there is a model that $\mathcal{T}$-satisfies it.

Context – Models and Satisfiability

- A model $\mathcal{A}$ satisfies ϕ iff ϕ evaluates to true under standard semantics
- For convenience, free variables are treated as uninterpreted constants.
- In SMT, for a given theory $\mathcal{T}$, a formula ϕ is $\mathcal{T}$-satisfied by a model $\mathcal{A}$ if $\mathcal{A}$ satisfies ϕ and it also satisfies $\mathcal{T}$.
- A formula is $\mathcal{T}$-satisfiable if there is a model that $\mathcal{T}$-satisfies it.
- $x < y < (x + 1)$ is $\mathcal{T}_{\mathsf{LRA}}$-satisfiable (linear real arithmetic) with a model $x = 0, y = .5$.

Context – Models and Satisfiability

- A model $\mathcal{A}$ satisfies ϕ iff ϕ evaluates to true under standard semantics
- For convenience, free variables are treated as uninterpreted constants.
- In SMT, for a given theory $\mathcal{T}$, a formula ϕ is $\mathcal{T}$-satisfied by a model $\mathcal{A}$ if $\mathcal{A}$ satisfies ϕ and it also satisfies $\mathcal{T}$.
- A formula is $\mathcal{T}$-satisfiable if there is a model that $\mathcal{T}$-satisfies it.
- $x < y < (x + 1)$ is $\mathcal{T}_{\mathsf{LRA}}$-satisfiable (linear real arithmetic) with a model $x = 0, y = .5$.
- $x < y < (x + 1)$ is $\mathcal{T}_{\mathsf{LIA}}$-unsatisfiable (linear integer arithmetic)

Context – Lazy solving

input : formula φ in theory $\mathcal{T}$
output: truth value

$\alpha \leftarrow \mathcal{T}2\mathcal{B}(\varphi)$ // abstract input formula
while true **do**
  $(\text{res}, \tau) \leftarrow \text{SAT}(\alpha)$ // Boolean model
  if res = false **then return** false
  $\mathcal{L} \leftarrow \bigcup_{l \in \tau} \mathcal{B}2\mathcal{T}(l)$ // convert to theory model
  $(\text{res}, \mathcal{L}') \leftarrow \mathcal{T}\text{-SAT}(\mathcal{L})$ // theory check
  if res = true **then return** true
  $\alpha \leftarrow \alpha \wedge \bigvee_{l \in \mathcal{L}'} \neg \mathcal{T}2\mathcal{B}(l)$ // block explanation
end

Observe: Effectively enumerating sets of literals $\mathcal{L}$, s.t. $\mathcal{L} \Rightarrow \varphi$.
Theory solver decides if $\mathcal{L}$ satisfiable.

Theory Combinations

- Until now we have assumed there is a single theory.

Theory Combinations

- Until now we have assumed there is a single theory.
- In practice important to combine theories.

$$(a = b+2) \wedge (A = write(B, a+1, 4)) \wedge (read(A, b+3) = 2 \vee f(a-1) \neq f(b+1))$$

Theory Combinations

- Until now we have assumed there is a single theory.
- In practice important to combine theories.

$$(a = b+2) \wedge (A = write(B, a+1, 4)) \wedge (read(A, b+3) = 2 \vee f(a-1) \neq f(b+1))$$

- Nelson-Oppen method: Use lazy solving and let the theory solvers communicate through equality.

Theory Combinations

- Until now we have assumed there is a single theory.
- In practice important to combine theories.

$$(a = b+2) \wedge (A = write(B, a+1, 4)) \wedge (read(A, b+3) = 2 \vee f(a-1) \neq f(b+1))$$

- Nelson-Oppen method: Use lazy solving and let the theory solvers communicate through equality.
- Setup: The theories $\mathcal{T}_1$ and $\mathcal{T}_2$ have the respective signatures: $\Sigma^1 = \Sigma^1_P \cup \Sigma^1_F \cup C$ and $\Sigma^2 = \Sigma^2_P \cup \Sigma^2_F \cup C$.

Theory Combinations

- Until now we have assumed there is a single theory.
- In practice important to combine theories.

$$(a = b+2) \wedge (A = write(B, a+1, 4)) \wedge (read(A, b+3) = 2 \vee f(a-1) \neq f(b+1))$$

- Nelson-Oppen method: Use lazy solving and let the theory solvers communicate through equality.
- Setup: The theories $\mathcal{T}_1$ and $\mathcal{T}_2$ have the respective signatures: $\Sigma^1 = \Sigma_P^1 \cup \Sigma_F^1 \cup C$ and $\Sigma^2 = \Sigma_P^2 \cup \Sigma_F^2 \cup C$.
- C is a set of constants and $\Sigma^1 \cap \Sigma^2 = C \cup \{=\}$.

Theory Combinations

- Until now we have assumed there is a single theory.
- In practice important to combine theories.

$$(a = b+2) \wedge (A = write(B, a+1, 4)) \wedge (read(A, b+3) = 2 \vee f(a-1) \neq f(b+1))$$

- Nelson-Oppen method: Use lazy solving and let the theory solvers communicate through equality.
- Setup: The theories $\mathcal{T}_1$ and $\mathcal{T}_2$ have the respective signatures: $\Sigma^1 = \Sigma^1_P \cup \Sigma^1_F \cup C$ and $\Sigma^2 = \Sigma^2_P \cup \Sigma^2_F \cup C$.
- C is a set of constants and $\Sigma^1 \cap \Sigma^2 = C \cup \{=\}$.
- Only equalities on C will be used to interact between the two respective theory solvers.

Theory Combinations

- Until now we have assumed there is a single theory.
- In practice important to combine theories.

$$(a = b+2) \wedge (A = write(B, a+1, 4)) \wedge (read(A, b+3) = 2 \vee f(a-1) \neq f(b+1))$$

- Nelson-Oppen method: Use lazy solving and let the theory solvers communicate through equality.
- Setup: The theories $\mathcal{T}_1$ and $\mathcal{T}_2$ have the respective signatures: $\Sigma^1 = \Sigma^1_P \cup \Sigma^1_F \cup C$ and $\Sigma^2 = \Sigma^2_P \cup \Sigma^2_F \cup C$.
- C is a set of constants and $\Sigma^1 \cap \Sigma^2 = C \cup \{=\}$.
- Only equalities on C will be used to interact between the two respective theory solvers.
- We want to decide a set of literals φ on the signature $\Sigma^1 \cup \Sigma^2$.

First Step: Purification

- Iteratively replace each sub-term t by a fresh constant $c \in C$ and add the literal $c = t$.

First Step: Purification

- Iteratively replace each sub-term t by a fresh constant $c \in C$ and add the literal $c = t$.
- Separate into two sets of literals φ_1, φ_2, so that φ_i is on the signature Σ^i.

First Step: Purification

- Iteratively replace each sub-term t by a fresh constant $c \in C$ and add the literal $c = t$.
- Separate into two sets of literals φ_1, φ_2, so that φ_i is on the signature Σ^i.

Example

First Step: Purification

- Iteratively replace each sub-term t by a fresh constant $c \in C$ and add the literal $c = t$.
- Separate into two sets of literals φ_1, φ_2, so that φ_i is on the signature Σ^i.

Example

- $\mathcal{T}_{\mathrm{E}}$...theory of the equality (EUF), $\mathcal{T}_{\mathbb{Z}}$...theory of integers.

First Step: Purification

- Iteratively replace each sub-term t by a fresh constant $c \in C$ and add the literal $c = t$.
- Separate into two sets of literals φ_1, φ_2, so that φ_i is on the signature Σ^i.

Example

- $\mathcal{T}_{\mathrm{E}}$...theory of the equality (EUF), $\mathcal{T}_{\mathbb{Z}}$...theory of integers.
- $1 \leq x \wedge x \leq 2 \wedge f(x) \neq f(1) \wedge f(x) \neq f(2)$

First Step: Purification

- Iteratively replace each sub-term t by a fresh constant $c \in C$ and add the literal $c = t$.
- Separate into two sets of literals φ_1, φ_2, so that φ_i is on the signature Σ^i.

Example

- $\mathcal{T}_{\mathrm{E}}$...theory of the equality (EUF), $\mathcal{T}_{\mathbb{Z}}$...theory of integers.
- $1 \leq x \wedge x \leq 2 \wedge f(x) \neq f(1) \wedge f(x) \neq f(2)$
- $w_1 \leq x \wedge x \leq w_2 \wedge f(x) \neq f(w_1) \wedge f(x) \neq f(w_2) \wedge w_1 = 1 \wedge w_2 = 2$

First Step: Purification

- Iteratively replace each sub-term t by a fresh constant $c \in C$ and add the literal $c = t$.
- Separate into two sets of literals φ_1, φ_2, so that φ_i is on the signature Σ^i.

Example

- $\mathcal{T}_{\mathrm{E}}$...theory of the equality (EUF), $\mathcal{T}_{\mathbb{Z}}$...theory of integers.
- $1 \leq x \wedge x \leq 2 \wedge f(x) \neq f(1) \wedge f(x) \neq f(2)$
- $w_1 \leq x \wedge x \leq w_2 \wedge f(x) \neq f(w_1) \wedge f(x) \neq f(w_2) \wedge w_1 = 1 \wedge w_2 = 2$
- $w_1 \leq x \wedge x \leq w_2 \wedge w_1 = 1 \wedge w_2 = 2$ and $f(x) \neq f(w_1) \wedge f(x) \neq f(w_2)$

First Step: Purification

- Iteratively replace each sub-term t by a fresh constant $c \in C$ and add the literal $c = t$.
- Separate into two sets of literals φ_1, φ_2, so that φ_i is on the signature Σ^i.

Example

- $\mathcal{T}_{\mathrm{E}}$...theory of the equality (EUF), $\mathcal{T}_{\mathbb{Z}}$...theory of integers.
- $1 \leq x \wedge x \leq 2 \wedge f(x) \neq f(1) \wedge f(x) \neq f(2)$
- $w_1 \leq x \wedge x \leq w_2 \wedge f(x) \neq f(w_1) \wedge f(x) \neq f(w_2) \wedge w_1 = 1 \wedge w_2 = 2$
- $w_1 \leq x \wedge x \leq w_2 \wedge w_1 = 1 \wedge w_2 = 2$ and
  $f(x) \neq f(w_1) \wedge f(x) \neq f(w_2)$
- Simplify $1 \leq x \wedge x \leq 2 \wedge w_1 = 1 \wedge w_2 = 2$ and
  $f(x) \neq f(w_1) \wedge f(x) \neq f(w_2)$

Second Step: Guess and Check

- For the separation φ_1, φ_2 consider all the shared constants X.

Second Step: Guess and Check

- For the separation φ_1, φ_2 consider all the shared constants X.
- Guess some equivalence relation R on X.

Second Step: Guess and Check

- For the separation φ_1, φ_2 consider all the shared constants X.
- Guess some equivalence relation R on X.
- Build the arrangement formula:

$$\alpha(X, R) = \bigwedge_{u,v \in X, uRv} u = v \ \wedge \bigwedge_{u,v \in X, \neg uRv} u \neq v$$

Second Step: Guess and Check

- For the separation φ_1, φ_2 consider all the shared constants X.
- Guess some equivalence relation R on X.
- Build the arrangement formula:

$$\alpha(X,R) = \bigwedge_{u,v \in X, uRv} u = v \ \wedge \bigwedge_{u,v \in X, \neg uRv} u \neq v$$

- Check satisfiability of $\varphi_1 \wedge \alpha(X,R)$ and $\varphi_2 \wedge \alpha(X,R)$.

Second Step: Guess and Check

- For the separation φ_1, φ_2 consider all the shared constants X.
- Guess some equivalence relation R on X.
- Build the arrangement formula:

$$\alpha(X, R) = \bigwedge_{u,v \in X, uRv} u = v \ \wedge \bigwedge_{u,v \in X, \neg uRv} u \neq v$$

- Check satisfiability of $\varphi_1 \wedge \alpha(X, R)$ and $\varphi_2 \wedge \alpha(X, R)$.
- $\varphi_1 \wedge \varphi_2$ is satisfiable iff the we get satisfiability for some R.

Guess and Check: Unsatisfiable Example

- $1 \leq x \land x \leq 2 \land w_1 = 1 \land w_2 = 2$ and $f(x) \neq f(w_1) \land f(x) \neq f(w_2)$

Guess and Check: Unsatisfiable Example

- $1 \leq x \wedge x \leq 2 \wedge w_1 = 1 \wedge w_2 = 2$ and $f(x) \neq f(w_1) \wedge f(x) \neq f(w_2)$
- Possible partitions: $\{\{x, w_1, w_2\}\}$ $\{\{x, w_1\}, \{w_2\}\}$ $\{\{x, w_2\}, \{w_1\}\}$ $\{\{x\}, \{w_1, w_2\}\}$ $\{\{x\}, \{w_1\}, \{w_2\}\}$

Guess and Check: Unsatisfiable Example

- $1 \leq x \wedge x \leq 2 \wedge w_1 = 1 \wedge w_2 = 2$ and $f(x) \neq f(w_1) \wedge f(x) \neq f(w_2)$
- Possible partitions: $\{\{x, w_1, w_2\}\}$ $\{\{x, w_1\}, \{w_2\}\}$ $\{\{x, w_2\}, \{w_1\}\}$ $\{\{x\}, \{w_1, w_2\}\}$ $\{\{x\}, \{w_1\}, \{w_2\}\}$
- $x = w_1 = w_2$ UNSAT in $\mathcal{T}_{\mathbb{Z}}$ because $w_1 = 1 \wedge w_2 = 2$.

Guess and Check: Unsatisfiable Example

- $1 \leq x \wedge x \leq 2 \wedge w_1 = 1 \wedge w_2 = 2$ and $f(x) \neq f(w_1) \wedge f(x) \neq f(w_2)$
- Possible partitions: $\{\{x, w_1, w_2\}\}$ $\{\{x, w_1\}, \{w_2\}\}$ $\{\{x, w_2\}, \{w_1\}\}$ $\{\{x\}, \{w_1, w_2\}\}$ $\{\{x\}, \{w_1\}, \{w_2\}\}$
- $x = w_1 = w_2$ UNSAT in $\mathcal{T}_{\mathbb{Z}}$ because $w_1 = 1 \wedge w_2 = 2$.
- $x = w_1 \wedge x \neq w_2$ UNSAT in $\mathcal{T}_{\mathrm{E}}$ because $f(x) \neq f(w_1)$.

Guess and Check: Unsatisfiable Example

- $1 \leq x \wedge x \leq 2 \wedge w_1 = 1 \wedge w_2 = 2$ and $f(x) \neq f(w_1) \wedge f(x) \neq f(w_2)$
- Possible partitions: $\{\{x, w_1, w_2\}\}$ $\{\{x, w_1\}, \{w_2\}\}$ $\{\{x, w_2\}, \{w_1\}\}$ $\{\{x\}, \{w_1, w_2\}\}$ $\{\{x\}, \{w_1\}, \{w_2\}\}$
- $x = w_1 = w_2$ UNSAT in $\mathcal{T}_{\mathbb{Z}}$ because $w_1 = 1 \wedge w_2 = 2$.
- $x = w_1 \wedge x \neq w_2$ UNSAT in $\mathcal{T}_{\mathrm{E}}$ because $f(x) \neq f(w_1)$.
- $x = w_2 \wedge x \neq w_1$ UNSAT in $\mathcal{T}_{\mathrm{E}}$ because $f(x) \neq f(w_2)$.

Guess and Check: Unsatisfiable Example

- $1 \leq x \land x \leq 2 \land w_1 = 1 \land w_2 = 2$ and $f(x) \neq f(w_1) \land f(x) \neq f(w_2)$
- Possible partitions: $\{\{x, w_1, w_2\}\}$ $\{\{x, w_1\}, \{w_2\}\}$ $\{\{x, w_2\}, \{w_1\}\}$ $\{\{x\}, \{w_1, w_2\}\}$ $\{\{x\}, \{w_1\}, \{w_2\}\}$
- $x = w_1 = w_2$ UNSAT in $\mathcal{T}_{\mathbb{Z}}$ because $w_1 = 1 \land w_2 = 2$.
- $x = w_1 \land x \neq w_2$ UNSAT in $\mathcal{T}_{\mathrm{E}}$ because $f(x) \neq f(w_1)$.
- $x = w_2 \land x \neq w_1$ UNSAT in $\mathcal{T}_{\mathrm{E}}$ because $f(x) \neq f(w_2)$.
- $x \neq w_1 \land w_1 = w_2$ UNSAT in $\mathcal{T}_{\mathbb{Z}}$ because $w_1 = 1 \land w_2 = 2$.

Guess and Check: Unsatisfiable Example

- $1 \leq x \wedge x \leq 2 \wedge w_1 = 1 \wedge w_2 = 2$ and $f(x) \neq f(w_1) \wedge f(x) \neq f(w_2)$
- Possible partitions: $\{\{x, w_1, w_2\}\}$ $\{\{x, w_1\}, \{w_2\}\}$ $\{\{x, w_2\}, \{w_1\}\}$ $\{\{x\}, \{w_1, w_2\}\}$ $\{\{x\}, \{w_1\}, \{w_2\}\}$
- $x = w_1 = w_2$ UNSAT in $\mathcal{T}_{\mathbb{Z}}$ because $w_1 = 1 \wedge w_2 = 2$.
- $x = w_1 \wedge x \neq w_2$ UNSAT in $\mathcal{T}_{\mathrm{E}}$ because $f(x) \neq f(w_1)$.
- $x = w_2 \wedge x \neq w_1$ UNSAT in $\mathcal{T}_{\mathrm{E}}$ because $f(x) \neq f(w_2)$.
- $x \neq w_1 \wedge w_1 = w_2$ UNSAT in $\mathcal{T}_{\mathbb{Z}}$ because $w_1 = 1 \wedge w_2 = 2$.
- $x \neq w_1 \wedge x \neq w_2 \wedge w_1 \neq w_2$ UNSAT in $\mathcal{T}_{\mathbb{Z}}$ because it cannot be that $x \neq 1 \wedge x \neq 2 \wedge 1 \leq x \leq 2$.

Guess and Check: Satisfiable Example

- $f(x) = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge f(x) \neq f(2)$.

Guess and Check: Satisfiable Example

- $f(x) = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge f(x) \neq f(2)$.
- $w_1 = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$

Guess and Check: Satisfiable Example

- $f(x) = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge f(x) \neq f(2)$.
- $w_1 = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$
- Shared: x, w_1, w_2.

Guess and Check: Satisfiable Example

- $f(x) = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge f(x) \neq f(2)$.
- $w_1 = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$
- Shared: x, w_1, w_2.
- Simplify as $w_1 = x + 1 \wedge x \leq 1 + z \wedge x + z \leq 1 \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$

Guess and Check: Satisfiable Example

- $f(x) = x + y \land x \leq y + z \land x + z \leq y \land y = 1 \land f(x) \neq f(2)$.
- $w_1 = x + y \land x \leq y + z \land x + z \leq y \land y = 1 \land w_2 = 2$ and $w_1 = f(x) \land f(x) \neq f(w_2)$
- Shared: x, w_1, w_2.
- Simplify as $w_1 = x + 1 \land x \leq 1 + z \land x + z \leq 1 \land y = 1 \land w_2 = 2$ and $w_1 = f(x) \land f(x) \neq f(w_2)$
- $x = w_1$ gives contradiction with $w_1 = x + 1$.

Guess and Check: Satisfiable Example

- $f(x) = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge f(x) \neq f(2)$.
- $w_1 = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$
- Shared: x, w_1, w_2.
- Simplify as $w_1 = x + 1 \wedge x \leq 1 + z \wedge x + z \leq 1 \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$
- $x = w_1$ gives contradiction with $w_1 = x + 1$.
- $x = w_2$ gives contradiction with $f(x) \neq f(w_2)$.

Guess and Check: Satisfiable Example

- $f(x) = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge f(x) \neq f(2)$.
- $w_1 = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$
- Shared: x, w_1, w_2.
- Simplify as $w_1 = x + 1 \wedge x \leq 1 + z \wedge x + z \leq 1 \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$
- $x = w_1$ gives contradiction with $w_1 = x + 1$.
- $x = w_2$ gives contradiction with $f(x) \neq f(w_2)$.
- x has to be in its own equivalence class, i.e. $x \neq w_1$, $x \neq w_2$.

Guess and Check: Satisfiable Example

- $f(x) = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge f(x) \neq f(2)$.
- $w_1 = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$
- Shared: x, w_1, w_2.
- Simplify as $w_1 = x + 1 \wedge x \leq 1 + z \wedge x + z \leq 1 \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$
- $x = w_1$ gives contradiction with $w_1 = x + 1$.
- $x = w_2$ gives contradiction with $f(x) \neq f(w_2)$.
- x has to be in its own equivalence class, i.e. $x \neq w_1$, $x \neq w_2$.
- Adding $w_1 = w_2$ permits models:
  $\{x = 1, z = 0, y = 1, w_1 = w_2 = 2\}$
  $\{w_1 = w_2 = *_1, x = *_2, f = \{*_1 \mapsto *_2, *_2 \mapsto *_1\}\}$

Guess and Check: Satisfiable Example

- $f(x) = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge f(x) \neq f(2)$.
- $w_1 = x + y \wedge x \leq y + z \wedge x + z \leq y \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$
- Shared: x, w_1, w_2.
- Simplify as $w_1 = x + 1 \wedge x \leq 1 + z \wedge x + z \leq 1 \wedge y = 1 \wedge w_2 = 2$ and $w_1 = f(x) \wedge f(x) \neq f(w_2)$
- $x = w_1$ gives contradiction with $w_1 = x + 1$.
- $x = w_2$ gives contradiction with $f(x) \neq f(w_2)$.
- x has to be in its own equivalence class, i.e. $x \neq w_1$, $x \neq w_2$.
- Adding $w_1 = w_2$ permits models:
  $\{x = 1, z = 0, y = 1, w_1 = w_2 = 2\}$
  $\{w_1 = w_2 = *_1, x = *_2, f = \{*_1 \mapsto *_2, *_2 \mapsto *_1\}\}$
- Merged model
  $\{x = y = 1, z = 0, w_1 = w_2 = 2, f = \{2 \mapsto 1, 1 \mapsto 2, \text{rest arbitrary}\}\}$

Merging

- For the models $(A_1, \mathcal{I}_1)$ and $(A_2, \mathcal{I}_2)$ build a bijection h between A_2 and A_1 so that $h(\mathcal{I}_2(c)) = \mathcal{I}_1(c)$ for all shared constants c.

Merging

- For the models $(A_1, \mathcal{I}_1)$ and $(A_2, \mathcal{I}_2)$ build a bijection h between A_2 and A_1 so that $h(\mathcal{I}_2(c)) = \mathcal{I}_1(c)$ for all shared constants c.
- Such h exists because both models must respect the guessed equivalence relation.

Merging

- For the models $(A_1, \mathcal{I}_1)$ and $(A_2, \mathcal{I}_2)$ build a bijection h between A_2 and A_1 so that $h(\mathcal{I}_2(c)) = \mathcal{I}_1(c)$ for all shared constants c.
- Such h exists because both models must respect the guessed equivalence relation.

Example

Merging

- For the models $(A_1, \mathcal{I}_1)$ and $(A_2, \mathcal{I}_2)$ build a bijection h between A_2 and A_1 so that $h(\mathcal{I}_2(c)) = \mathcal{I}_1(c)$ for all shared constants c.
- Such h exists because both models must respect the guessed equivalence relation.

Example

- Models: $\mathcal{I}_1 = \{x = 1, z = 0, y = 1, w_1 = w_2 = 2\}$
  $\mathcal{I}_2 = \{w_1 = w_2 = *_1, x = *_2, f = \{*_1 \mapsto *_2, *_2 \mapsto *_1\}\}$

Merging

- For the models $(A_1, \mathcal{I}_1)$ and $(A_2, \mathcal{I}_2)$ build a bijection h between A_2 and A_1 so that $h(\mathcal{I}_2(c)) = \mathcal{I}_1(c)$ for all shared constants c.
- Such h exists because both models must respect the guessed equivalence relation.

Example

- Models: $\mathcal{I}_1 = \{x = 1, z = 0, y = 1, w_1 = w_2 = 2\}$
  $\mathcal{I}_2 = \{w_1 = w_2 = *_1, x = *_2, f = \{*_1 \mapsto *_2, *_2 \mapsto *_1\}\}$
- $h(*_1) = 2$, $h(*_2) = 1$, rest some arbitrary bijection.

Nelson-Oppen: When does it work?

- Nelson-Oppen does not to work for arbitrary theories.

Nelson-Oppen: When does it work?

- Nelson-Oppen does not to work for arbitrary theories.
- Theories needs to be stably infinite

Nelson-Oppen: When does it work?

- Nelson-Oppen does not to work for arbitrary theories.
- Theories needs to be stably infinite
  - A theory $\mathcal{T}$ is stably infinite iff for every satisfiable $\Sigma_{\mathcal{T}}$-formula ϕ, there is a $\mathcal{T}$-model that satisfies ϕ and that has a universe of infinite cardinality

Nelson-Oppen: When does it work?

- Nelson-Oppen does not to work for arbitrary theories.
- Theories needs to be stably infinite
  - A theory $\mathcal{T}$ is stably infinite iff for every satisfiable $\Sigma_{\mathcal{T}}$-formula ϕ, there is a $\mathcal{T}$-model that satisfies ϕ and that has a universe of infinite cardinality
  - Examples: EUF, LIA, LRA, etc.

Nelson-Oppen: When does it work?

- Nelson-Oppen does not to work for arbitrary theories.
- Theories needs to be stably infinite
  - A theory $\mathcal{T}$ is stably infinite iff for every satisfiable $\Sigma_{\mathcal{T}}$-formula ϕ, there is a $\mathcal{T}$-model that satisfies ϕ and that has a universe of infinite cardinality
  - Examples: EUF, LIA, LRA, etc.
- Combination of more than two theories is done analogously.

Nelson-Oppen: When does it work?

- Nelson-Oppen does not to work for arbitrary theories.
- Theories needs to be stably infinite
  - A theory $\mathcal{T}$ is stably infinite iff for every satisfiable $\Sigma_{\mathcal{T}}$-formula ϕ, there is a $\mathcal{T}$-model that satisfies ϕ and that has a universe of infinite cardinality
  - Examples: EUF, LIA, LRA, etc.
- Combination of more than two theories is done analogously.
- For certain theories there are more efficient methods than considering all partitions (convex theories).

Summary

- Theory combinations:

Summary

- Theory combinations:
  - Nelson-Oppen method enables combining theory solvers (under certain conditions).

Summary

- Theory combinations:
  - Nelson-Oppen method enables combining theory solvers (under certain conditions).
  - Theory solvers communicate through equality.