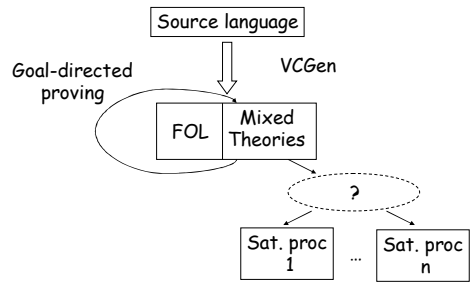
Nelson-Oppen Architecture for Cooperating Decision Procedures

Review



Review: Theories

- A theory has
  - Syntax: a set of n-ary constants
  - Axioms: a number of axioms
- Satisfiability problem: decide whether a conjunction of literals from the theory is satisfiable
- F is satisfiable when
  - Exists a universe (set) U
  - and an interpretation Ψ of variables and constants such that
    - Ψ satisfies the axioms of the theory
    - Ψ satisfies F
- E.g., $x \geq y \wedge x + y \geq 0$ is satisfiable
  - U is $\mathbb{Z}$, $\Psi = \{ x = 0_{\mathbb{Z}}, y = 0_{\mathbb{Z}}, 0 = 0_{\mathbb{Z}}, + = +_{\mathbb{Z}}, \geq = \geq_{\mathbb{Z}}, \ldots \}$

Combining Satisfiability Procedures

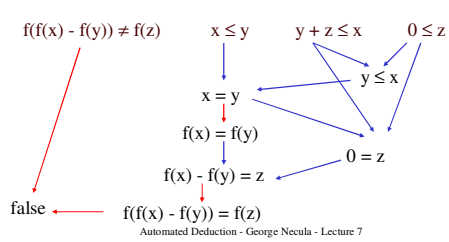
- Consider a set of literals F
  - Containing symbols from two theories T_1 and T_2
- We split F into two sets of literals
  - F_1 containing only literals in theory T_1
  - F_2 containing only literals in theory T_2
  - We name all subexpressions:
    $p_1(f_2(E))$ is split into $f_2(E) = n \wedge p_1(n)$
- We have: unsat $(F_1 \wedge F_2)$ iff unsat(F)
  - Also, unsat$(F_1) \vee$ unsat$(F_2) \Rightarrow$ unsat(F)
  - But the converse is not true
- So we cannot compute unsat(F) with a trivial combination of the sat. procedures for T_1 and T_2

Example: Satisfiability of Mixed Theories

- Again: $F = f(f(x) - f(y)) \neq f(z), x \leq y, y + z \leq x, z \geq 0$
- Separate $F = F_E \wedge F_R$ such that
  F is sat iff $F_R \wedge F_E$ is sat
  - F_E (from E) = $g_1 = f(x), g_2 = f(y), f(g_3) = f(z)$
  - F_R (from R) = $x \leq y, y + z \leq x, z \geq 0, g_3 = g_1 - g_2$
- Both F_E and F_R are independently satisfiable
  - But their conjunction is not

Combining Satisfiability Procedures. Example

- Consider equality and arithmetic



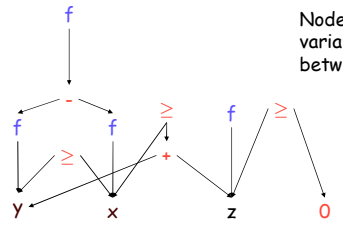
Combining Satisfiability Procedures

- Combining satisfiability procedures is non trivial
- And that is to be expected:
  - Equality was solved by Ackerman in 1924, arithmetic by Fourier even before, but E + A only in 1979 !
- Yet in any single verification problem we will have literals from several theories:
  - Equality, arithmetic, lists, …
- When and how can we combine separate satisfiability procedures?

Nelson-Oppen Method (1)

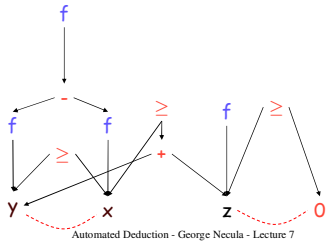
1. Represent all conjuncts in the same DAG
   $f(f(x) - f(y)) \neq f(z) \wedge y \geq x \wedge x \geq y + z \wedge z \geq 0$

Node names act as variables shared between theories



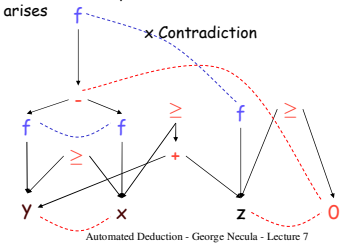
Nelson-Oppen Method (2)

2. Run each sat. procedure
   - Require it to report all contradictions (as usual)
   - Must also report all equalities between nodes (key idea)



Nelson-Oppen Method (3)

3. Broadcast all discovered equalities and re-run sat. procedures
   - Until no more equalities are discovered or a contradiction arises



Nelson-Oppen Method (4)

- If a contradiction is ever found then unsatisfiable
  - This is clearly sound, if sat procedures are sound
  - … because only sound equalities are ever found
- If there are no more equalities to be found by either sat proc then we report satisfiable
  - Is this complete ? Have they exchanged enough info ?
  - The two sat procs have each found satisfying assignments. how do we know that they are ecompatible?

Completeness

- Let F_1 in theory T_1 (with axioms A_1) and F_2 in theory T_2 (with axioms A_2) such that:
  - F_1, A_1 is satisfiable
  - F_2, A_2 is satisfiable
  - $\forall xy \in Var.\ F_1, A_1 \vdash x = y$ iff $F_2, A_2 \vdash x = y$
    - They share all equalities between variables that each can infer
    - If this is not true, we add $x=y$ to either F_1 or F_2

  then $F_1 \wedge F_2$, $A_1 \wedge A_2$ is satisfiable
- We'll discover more conditions that must hold for this to be true
  - We do that by looking at the proof

Completeness proof

- F_1, A_1 is satisfiable
  - There exists U_1, Ψ_1 that satisfies F_1, A_1
  - Similarly there exists U_2, Ψ_2 that satisfies F_2, A_2
- We will extend Ψ_1 to Ψ : Terms $\rightarrow U_1$ such that
  $\Psi(t)$ = if $t \in T_1$ then $\Psi_1(t)$ else $\alpha(\Psi_2(t))$
  - Where $\alpha : U_2 \rightarrow U_1$ a bijection
  - We need $\Psi(t) = \Psi(t')$ iff $\Psi_2(t) = \Psi_2(t')$ (α injective)
  - Also must have $(\Psi_2 \models \forall x.\ A) \Rightarrow (\Psi \models \forall x.\ A)$ (α onto)
  - Ψ is an interpretation that satisfies $F_1 \wedge F_2$
- This means that U_1 and U_2 must have the same cardinality !
  - Typically, countable and infinite sets

Theories Admit Isomorphic Models

- Condition 1 (sufficient for isomorphism):
  Theories must have countable and infinite models
- Conterexample:
  - Theory1: $\exists a,b\ \forall x.\ x = a \vee x = b$
  - Theory2: $\exists a,b,c.\ a \neq b \wedge b \neq c \wedge a \neq c$
  - Empty conjunction is unsat, but Nelson-Oppen cannot prove it
  - The theories do not have isomorphic models !

More on Isomorphism

- Variables belong to both theories:
  $\Psi(x) = \Psi_1(x) = \alpha(\Psi_2(x))$
- We need interpretations such that
  $\forall xy \in Var.\ \Psi_1(x) = \Psi_1(y)$ iff $\Psi_2(x) = \Psi_2(y)$
- Sufficient conditions:
  $\forall\ xy \in Var.\ \Psi_1(x) = \Psi_1(y)$ iff $F_1, A_1 \vdash x = y$ (also in T_2)
  - Because already have $F_1, A_1 \vdash x = y$ iff $F_2, A_2 \vdash x = y$
- Means that there must be an interpretation that makes equal ONLY variables that MUST be equal

Necessary Interpretations

- Means that there must be an interpretation that makes equal ONLY variables that MUST be equal
  - Call this a "necessary" interpretation
- Example: $F = x \geq 0,\ x + y \geq 0$
  - $\Psi = \{ x = 0, y = 0\}$ not a necessary interpretation
  - There are necessary ones: $\Psi = \{ x = 0, y = 1\}$
  - Which theories always have necessary interpretations ?

Necessary Interpretations

- Consider theory T with axioms A and literals F
- Define the equivalence class of x
  $Class_x = \{ y \mid F, A \vdash x = y \}$
- Let x_i be the representatives
- We must find Ψ such that $\Psi(x_i) \neq \Psi(x_j)$ if $i \neq j$
- Need:

if F is satisfiable,

then $F \wedge \bigwedge_{i \neq j} x_i \neq x_j$ must also be satisfiable

Convex Theories

- A theory is <u>convex</u> if whenever a satisfiable conjunction of literals entails a disjunction of equalities of variables, then it entails one of the equalities.
- Example:
  - Theory of linear arithmetic with equalities
  - F is a conjunction of literals with three variables (x, y, z)
  - F denotes a 3D point, a line, a plane, or the whole space
  - Say $F \Rightarrow x = y \vee x = z$
  - It must be the case that F is all inside $x = y$ or all inside $x = z$

Convex Theories Have Necessary Interp.

- If T is convex, then $F \wedge \wedge_{i \neq j} x_i \neq x_j$ is satisfiable
  - Otherwise, $F, A \vdash \vee_{i \neq j} x_i = x_j$
  - and thus $F, A \vdash x_i = x_j$ (because T is convex). Contradiction
- Example: Consider $(\mathbb{Z}, +, \leq)$ and Equality
  - Consider: $1 \leq x \leq 2 \wedge a = 1 \wedge b = 2 \wedge f(x) \neq f(a) \wedge f(x) \neq f(b)$
  - No equalities and no contradictions are discovered
  - Yet, unsatisfiable
  - Nelson-Oppen does not work here (theory is non-convex)

Homework

- Give another example of a non-convex theory (other than sel/upd and the example from previous slide)
  - Relatively easy
- Show that if all the axioms of the theory are of the form $\forall x. f_1(f_2(\ldots f_n(x))) = g_1(g_2(\ldots g_m(x)))$, then the theory is convex
- Is this still true if we allow disequalities also in the axioms?

Finish Completeness Proof

- We have bijection α working on variables
  - Convexity ensures that
- Define $\Psi(f) = \lambda x_1 \ldots x_n.\ \alpha(\Psi_2(f)(\alpha^{-1}(x_1), \ldots, \alpha^{-1}(x_n))$
  - if $f \in T_2$
  - and $\Psi(f) = \Psi_1(f)$ if $f \in T_1$
- For this to be possible the theories cannot have common function symbols
- Example: $\mathbb{Z}^+$ and $\mathbb{R}^{+,*}$ are decidable but $\mathbb{Z}^{+,*}$ is undecidable
  - No hope for Nelson-Oppen to work for the combination

Applicability of Nelson-Oppen

1. Theories must have disjoint sets of function symbols
2. Theories must admit countably infinite models
3. Theories must be convex
   - the biggest obstacle in practice

Another formulation of Nelson-Oppen

Facts ::= Set of literals

```
NO (F : Facts) =
  match asat(F), bsat(F) with
    | Contra, _ -> true
    | _, Contra -> true
    | Eq x =y, _ -> NO (F ∪ { x = y } )
    | _, Eq x = y -> NO (F ∪ { x = y })
    | Sat, Sat    -> raise NoProof
```

- Soundness (let F_0 be the original set of facts):
  - Precondition for NO F is $F_0 \Rightarrow F$
  - Postcondition if result is true: $F \Rightarrow \bot$
  - Therefore: $F_0 \Rightarrow \bot$

Soundness Proof for N-O

- We assume that
  - asat(F) = Contra only if $F \Rightarrow \bot$
  - asat(F) = "Eq x = y" only if $F \Rightarrow x = y$
- Core of the soundness proof
  - If asat(F) = Contra
    - $F \Rightarrow \bot$, q.e.d
  - If asat(F) = Eq x = y
    - Then $F \Rightarrow x = y$
    - We meet the invariant $F_0 \Rightarrow F \cup \{ x = y \})$
    - Assume no $(F \cup \{ x = y \})$ = true
    - Thus $F \wedge x = y \Rightarrow \bot$
    - Thus $F \Rightarrow \bot$

Proof Generation

- We want to change NO to emit proofs
  - No need to trust the prover
  - Can find bugs in the prover !
  - Can use for proof-carrying code
- Essentially, we implement the soundness argument
  - On every run, a soundness proof is constructed

Proof Representation

- Proofs are trees
  - Leaves are the hypotheses or axioms
  - Internal nodes are instances of inference rules
- Example:
  - proof of $\forall x \forall y.\ x = y \Rightarrow (y = x \wedge \exists z.\ x = z)$
- We use a number of constructors
  - Nullary constructors for axioms
  - N-ary constructors for inference rules with N hypotheses

Proof Representation

- Axiom: "true introduction"
  - Constant truei : pf
  - pf is the type of proofs

$$\frac{}{\vdash \text{true}}$$

- Inference rule: "conjunction introduction"
  - Constant: andi: pf → pf → pf

$$\frac{\vdash A \qquad \vdash B}{\vdash A \wedge B}$$

- Inference rule: "conjunction elimination"
  - Constant: andel: pf → pf

$$\frac{\vdash A \wedge B}{\vdash A}$$

- Problem:
  andel truei : pf but is not a representation of a valid proof
  - We need a more expressive type system to encode proofs such that well-typed encodings represent valid proofs

Dependent Types

- We make pf a type family indexed by formulas
  - f : Type (the type of encodings of formulas)
  - e : Type (the type of encodings of expressions)
  - pf : f → Type (the type of proofs indexed by formulas)
- Examples:
  - true : f
  - and: f → f → f
  - 0 : e
  - truei : pf true
  - andi : pf A → pf B → pf (and A B)
    - We need to bind A and B somewhere

Dependent Types

- $\Pi x{:}T.\ T'$
  - Type of a function with argument of type T and result of type T', where the <u>value</u> of the argument (x) might be referred to in the type of the result (T')
- Example
  andi: ΠA:f. ΠB:f. pf A → pf B → pf (and A B)
- Typing rules for λ-calculus with depent types (λ^{Π})

$$\frac{\Gamma, x : \tau_2 \vdash e : \tau}{\Gamma \vdash \lambda x : \tau_2.e : \Pi x : \tau_2.\tau} \qquad \frac{\Gamma \vdash e_1 : \Pi x : \tau_2.\tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : [e_2/x]\tau}$$

Proof Checking

- Now we can validate proof representations by type-checking them
- Example:
  - Proof of $A \wedge B$ from the assumptions A and B
  - Must check
    $A : f, B : f, D_A : pf\ A, D_B : pf\ B \vdash andi\ A\ B\ D_A\ D_B : pf\ (and\ A\ B)$
  - You can verify that this holds indeed in λ^{Π} with the right constant declarations
- Thus <u>Type Checking = Proof Checking</u>

Hypothetical Judgments

- Recall "implication introduction"
- We add constants
  - impl: f → f → f
  - impli: ΠA:f.ΠB:f. ?? → pf (impl A B)
  - We cannot simply replace ?? by pf B
- Also, the assumption ⊢ A must be local !
  - E.g., the following is not a valid proof of (A ⇒ A) ∧ A

$$\frac{\vdash B}{\vdash A \Rightarrow B}\ \begin{array}{c}\vdash A\\ \dots\end{array}$$

$$\frac{\dfrac{\begin{array}{c}\vdash A\\ \dots\end{array}}{\vdash A \Rightarrow A} \qquad A}{\vdash (A \Rightarrow A) \wedge A}$$

Hypothetical Judgments

- Proof are not simple trees
  - There are side-conditions for vaility
  - But we can represent these in the type system ?
- Need a notion of "local assumption"
- We'll use functions (parameters are local assumptions)
  - impli: ΠA:f.ΠB:f. (pf A → pf B) → pf (impl A B)
- Example: proof of A ⇒ A ∧ A

  impli A (and A A) (λx: pf A. andi A A x x)
- Proof representations are large
  - We will omit some of the Π-quantified arguments

  impli _ _ (λx:pf A. andi _ _ x x)

Parametric Judgments

- Recall "universal introduction"
  - another side-condition
  - But "fresh" means "local"

$$\frac{\vdash [a/x]A \quad (a \text{ is fresh})}{\vdash \forall x.A}$$

- We represent bound variables in the logic with bound variables in the meta-logic
  - all: (e → f). f
  - Example: ∀x. x = x represented as (all (λx. eq x x))
  - Note: ∀y. y = y has an α-equivalent representation !
  - Note: substitution is done by β-conversion in meta-logic

    [E/x] (x = x) is (λx. eq x x) E

Parametric Judgments

- Universal introduction

$$\frac{\vdash [a/x]A \quad (a \text{ is fresh})}{\vdash \forall x.A}$$

  alli: ΠA: (e → f). (Π a:e. pf (A a)) → pf (all A)
- Universal elimination

$$\frac{\vdash \forall x.A}{\vdash [E/x]A}$$

  alle: ΠA: (e → f). Π E:e. pf (all A) → pf (A E)
- Existential introduction

$$\frac{\vdash [E/x]A}{\vdash \exists x.A}$$

  existi: ΠA: (e → f). Π E:e. pf (A E) → pf (exists A)
- Existential elimination

  existe: ΠA: (e → f). Π B:f.
  pf (exists A) →
  (Πa:e. pf (A a) → pf B) → pf B

$$\frac{\vdash \exists x.A \qquad \begin{array}{c}\vdash [a/x]A\\ \dots\\ \vdash B\end{array}}{\vdash B}$$

Proof-By-Inversion with Proof Generation

- Recall the prove function

  prove: literal set → goal → bool
- Changes
  - We carry proofs with each literal
  - prove throws an exception if it cannot prove the goal

  prove: (pair of L:f and pf L) list → ΠG: goal. pf G

Proof-By-Inversion with Proof Generation

- prove H ($G_1 \wedge G_2$) =

  andi (prove H G_1) (prove H G_2)
- prove H (L ⇒ G) =

  impli (λh:pf L. prove (H ∪ { (L, h) }) G)
- prove H (∀x.G) =

  alli (λa:e. prove H ([a/x]G))
- prove H L =

  contra (λh:pf ¬ L. no (H ∪ { (¬ L, h)}))

$$\frac{\begin{array}{c}\vdash \neg L\\ \dots\\ \vdash \bot\end{array}}{\vdash L}$$

contra: ΠL: f. (pf (neg L) → pf false) → pf L

Nelson-Oppen with Proof Generation

NO: (pair of L:f and pf L) list → pf false

- NO F =

```
match asat(F), bsat(F) with
  | Contra d, _ -> d
  | _, Contra d -> d
  | Eq (x =y, d), _ -> NO (F ∪ { (x = y, d) } )
  | _, Eq (x = y, d) -> NO (F ∪ { (x = y, d) })
  | Sat, Sat    -> raise NoProof
```

- As long as
  - If asat F = Contra d, then d : pf false
  - If asat F = Eq (x = y, d) , then d : pf (eq x y)

Imperative Version of Nelson-Oppen

- Consider prove+NO for

  $L_1 \Rightarrow ((L_2 \Rightarrow L_3) \wedge (L_4 \Rightarrow L_5))$
- We invoke NO with the following arguments
  1. $L_1, L_2, \neg L_3$
  2. $L_1, L_4, \neg L_5$
- Each time we invoke asat with a growing set of facts
- Idea: have incremental satisfiability procedures
  - Have internal state
  - Support push/pop for literals
- Operations: push L_1; push L_2; push $\neg L_3$; pop; pop; push L_4; push $\neg L_5$; pop; pop

Requirements for Satisfiability Procedures

- Report all contradictions, along with pf false
- Report all equalities between variables, with proof
- Allow for incremental adding of literals
- Allow for retracting of literals
  - Take snapshots of the state
  - Or, revert to an equivalent state
  - This is a huge source of bugs
- Also, theory must be convex …

Handling Non-Convex Theories

- Many theories are non-convex
  - Theory of sel/upd

    true ⇒ x = y ∨ sel(upd(m,x,v),y) = sel(m,y)
  - Other examples, your homework
- For such theories it can be the case that
  - No contradiction is discovered
  - No single equality is discovered
  - But a disjunction of equalities is discovered
- We need to propagate disjunction of equalities …

Propagating Disjunctions of Equalities

- To propagate disjunctions we perform a case split:
- If a disjunction of equalities $E_1 \vee \ldots \vee E_n$ is discovered:
  - Must try to derive a contradiction for <u>each</u> E_i assumption !
- NO F =

```
match asat(F), bsat(F) with
  …
  | Disj (x1 = y1 ∨ x2 = y2, d) →
      let di = λhi: pf (eq xi yi). NO (F ∪ { (xi = yi, hi) })
      ori d d1 d2
```

ore: Π A,B,C:f. pf (or A B) →
(pf A → pf C) → (pf B → pf C) → pf C

Handling Non-Convex Theories

- Case splitting is expensive
  - Must backtrack (performance --)
  - Must implement all satisfiability procedures in incremental fashion (simplicity --)
- In some cases the splitting can be prohibitive:
  - Take pointers for example.

  upd(upd(…(upd(m, i_1, x), …, i_{n-1}, x), i_n, x) =
  upd(…(upd(m, j_1, x), …, j_{n-1}, x) ∧
  sel(m, i_1) ≠ x ∧ … ∧ sel(m, i_n) ≠ x
  entails $\bigvee_{j \neq k}\ i_j \neq i_k$
  (a conjunction of length n entails n^2 disjuncts)