Lecture 18

Pete Manolios
Northeastern

Set of Support

- Partition T the input clauses into two disjoint sets, S, the *set of support* of T and the unsupported clauses U. Restrict U-resolution so that no two clauses in U are resolved together
- Theorem: Let T be an Unsat set of clauses and let S be a subset of T where $T \backslash S$ is Sat; then there is a U-resolution proof of Usat(T) with set of support S
- Idea: focus U-resolution on finding resolvents that contribute to the solution
- For example say A is a set of standard mathematical axioms
  - You want to prove $B \Rightarrow C$
  - Using U-resolution you will want to derive the empty clause from A, B, $\neg C$
  - Since Sat(A) you can choose B, $\neg C$ as the set of support
  - Since A, B are Sat (presumably), you can choose $\neg C$ as the set of support
  - Suppose $\neg C$ is the only negative clause, then similar to negative resolution, but negative resolution is more restrictive; however, set of support often makes up for this by finding shorter proofs

Universal Horn Formulas

- A formula is a *universal Horn formula* if it is logically equivalent to a conjunction of formulas of the following form, where φ, φ_i, are atomic

$\langle \forall x_1, \ldots, x_n\ \varphi \rangle$	positive	
$\langle \forall x_1, \ldots, x_n\ \varphi_1 \wedge \cdots \wedge \varphi_m \ \Rightarrow\ \varphi \rangle$	positive	
$\langle \forall x_1, \ldots, x_n\ \neg\varphi_1 \vee \cdots \vee \neg\varphi_m \rangle$	negative	

differs from positive resolution!
we'll use pos/neg in this sense during the lecture

- Let Φ be a set of universal Horn sentences s.t. Sat(Φ); let Φ^+ be the subset of positive sentences in Φ; let ψ_i be atomic over vars $x_1,\ldots,x_n$; then
  - $\Phi \models (\psi_0 \wedge \cdots \wedge \psi_k)\sigma$ iff $\Phi^+ \models (\psi_0 \wedge \cdots \wedge \psi_k)\sigma$ if $\psi_i\sigma$ is ground for all i
  - $\Phi \models \langle \exists x_1,\ldots,x_n\ \psi_0 \wedge \cdots \wedge \psi_k \rangle$ iff $\Phi^+ \models \langle \exists x_1,\ldots,x_n\ \psi_0 \wedge \cdots \wedge \psi_k \rangle$
- The above is a key insight that often allows us to restrict attention to positive universal Horn formulas
- For propositional logic, Sat for Horn formulas is in P!

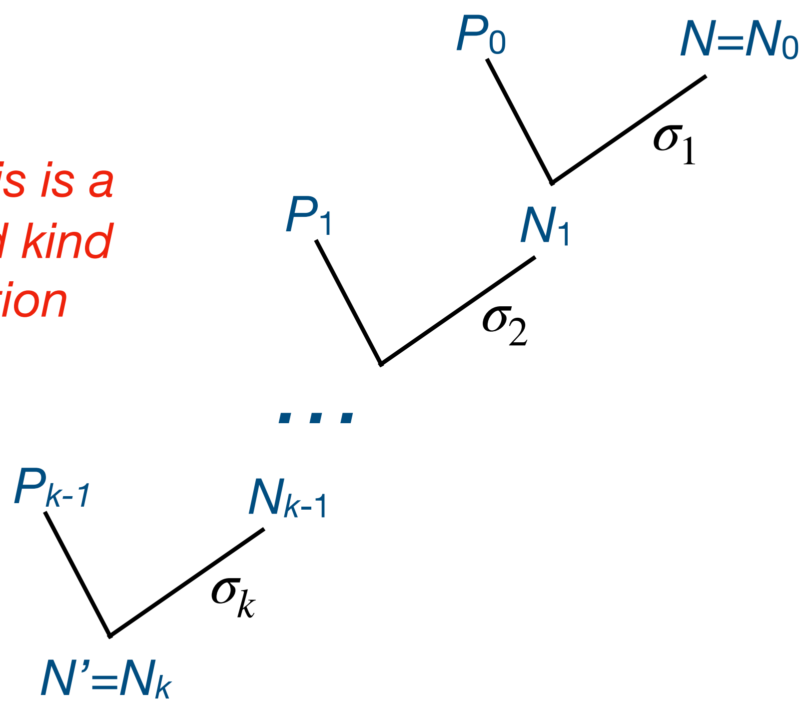
Free Models

- *Herbrand universe, H,* of FO language L is the set of all ground terms of L, except that if L has no constants, we add c to make the universe non-empty
- Let Φ be a set of universal Horn sentences over L s.t. Sat(Φ)
- There is $\mathscr{I}^\Phi$, an interpretation for Φ over *H* s.t. $\mathscr{I}^\Phi \models \phi$ iff $\Phi \models \phi$ for all atomic φ
  - Note: if $\Phi \models t_1{=}t_2$ then $\mathscr{I}^\Phi \models t_1{=}t_2$ *We include = here but we're still only considering checking FO w/out =*
  - Note: If $\Phi \models R(t_1, \ldots, t_n)$ then $\mathscr{I}^\Phi \models R(t_1, \ldots, t_n)$
  - Note: If neither $\Phi \models R(t_1, \ldots, t_n)$ nor $\Phi \models \neg R(t_1, \ldots, t_n)$ then $\mathscr{I}^\Phi \models \neg R(t_1, \ldots, t_2)$
  - So $\mathscr{I}^\Phi$, is *minimal (free)*: it only contains positive atomic information
  - There is a homomorphism between $\mathscr{I}^\Phi$ and any other model of Φ
- We have reduced $\Phi \models \phi$ to $\mathscr{I}^\Phi \models \phi$
  - Instead of checking if every interpretation of Φ satisfies φ
  - We only need to check a single, minimal interpretation
- Enables us to find solutions to queries in a systematic way
- Basis for logic programming

Logic Programming

- Let $\mathfrak{P}$ be a set of positive clauses and let N be a negative clause
  - A sequence $N_0, \ldots, N_k$ of negative clauses is a UH-resolution from $\mathfrak{P}$ and N iff $\exists\ P_0, \ldots, P_{k-1} \in \mathfrak{P}$ s.t. $N_0 = N$ and N_{i+1} is a U-resolvent of P_i and N_i for $i < k$
  - A negative clause N' is *UH-derivable from* $\mathfrak{P}$ *and* N iff $\exists$ a UH-resolution $N_0, \ldots, N_k$ from $\mathfrak{P}$ and N with $N' = N_k$

Notice that this is a very restricted kind of U-resolution



Logic Programming

- Let $\mathfrak{P}$ be a set of positive clauses and let N be a negative clause
  - A sequence $N_0, \ldots, N_k$ of negative clauses is a UH-resolution from $\mathfrak{P}$ and N iff $\exists\, P_0, \ldots, P_{k-1} \in \mathfrak{P}$ s.t. $N_0 = N$ and N_{i+1} is a U-resolvent of P_i and N_i for $i < k$
  - A negative clause N' is *UH-derivable from* $\mathfrak{P}$ *and* N iff $\exists$ a UH-resolution $N_0, \ldots, N_k$ from $\mathfrak{P}$ and N with $N'=N_k$
- Let $\mathcal{K}$ be a set of clauses, $\text{UHRes}(\mathcal{K})=\mathcal{K} \cup \{N \mid N$ is a negative clause and $\exists$ a positive/negative $P, N' \in \mathcal{K}$ s.t. N is a U-resolvent of P and $N'\}$
- $\text{UHRes}_0(\mathcal{K})=\mathcal{K}$
- $\text{UHRes}_{n+1}(\mathcal{K})=\text{UHRes}(\text{UHRes}_n(\mathcal{K}))$
- $\text{UHRes}_\omega(\mathcal{K})=\bigcup_{n\in\omega}\text{UHRes}_n(\mathcal{K})$

Standard recursive definition on the naturals

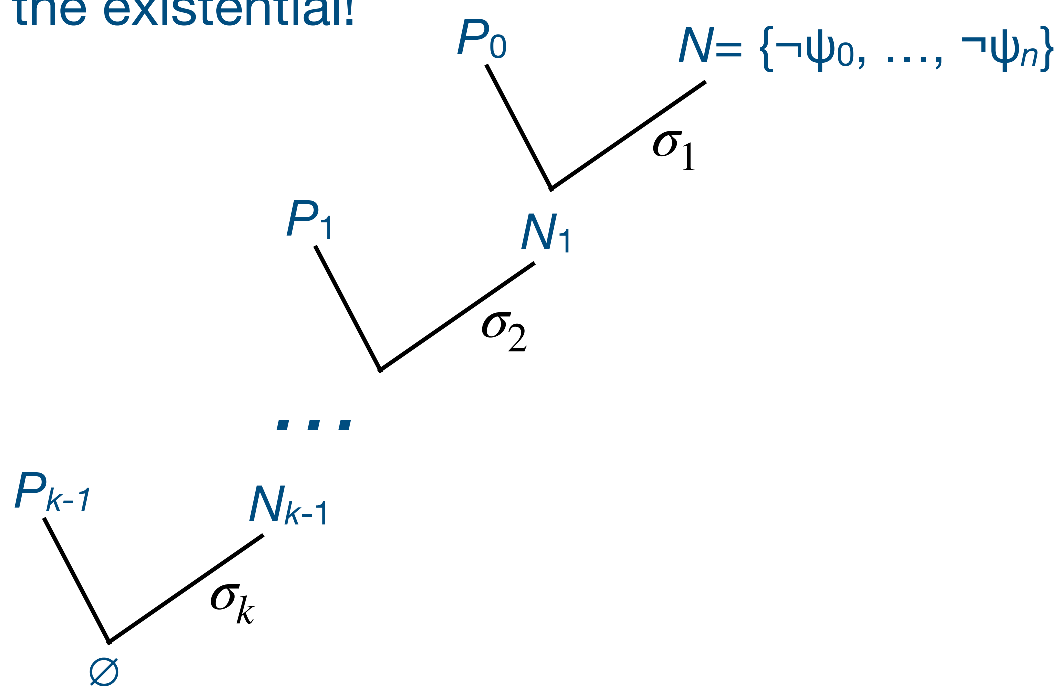
Standard recursive definition with limit ordinals

Logic Programming

Recall
$\mathcal{K}(\Phi)$=clauses of Φ

Theorem: Let Φ be a set of positive universal Horn sentences, $\mathfrak{P} = \mathcal{K}(\Phi)$, ψ_i atomic, $\langle \exists x_1,\ldots,x_n\ \psi_0 \wedge \cdots \wedge \psi_m \rangle$ a sentence and $N = \{\neg\psi_0, \ldots, \neg\psi_m\}$. Then:

- $\Phi \vDash \langle \exists x_1,\ldots,x_n\ \psi_0 \wedge \cdots \wedge \psi_m \rangle$ iff $\varnothing$ is UH-derivable from $\mathfrak{P}$ and N
- Given such a UH-derivation, with $\sigma_1, \ldots, \sigma_k$, $\Phi \vDash (\psi_0 \wedge \cdots \wedge \psi_m)\sigma_k\ldots\sigma_1$
- If $\Phi \vDash (\psi_0 \wedge \cdots \wedge \psi_m)\tau$, then there is a UH-derivation with $(\sigma_k\ldots\sigma_1) \leq \tau$
- So, we can find all solutions to the existential!



Logic Programming Example

$$\Phi = \{\langle \forall x, y\ P(x, y, c) \Rightarrow R(y, g(f(x)))\rangle, \langle \forall x, y\ P(f(x), y, c)\rangle\} \models \langle \exists x, y\ R(f(x), g(y))\rangle$$

$$\{\neg P(u, v, c), \underline{R(v, g(f(u)))}\} \quad \{\underline{\neg R(f(x), g(y))}\}$$

$$\sigma_1 = f(x), f(u) \leftarrow v, y$$

$$\{\underline{P(f(v), y, c)}\} \quad \{\underline{\neg P(u, f(x), c)}\}$$

$$\sigma_2 = f(x), f(v) \leftarrow y, u$$

$$\emptyset$$

- Recall: given a UH-derivation, with $\sigma_1, \ldots, \sigma_k$, $\Phi \models (\psi_0 \wedge \cdots \wedge \psi_m)\sigma_k \ldots \sigma_1$
- So, the following hold

$$\Phi \models R(f(x), g(f(f(v)))) \qquad \Phi \models \langle \forall x, v\ R(f(x), g(f(f(v))))\rangle$$

- And we have a family of solutions

Prolog

- One of the most popular logic programming languages is Prolog
- Given a set of Horn clauses and a query, find solutions

This is implication, ie, X :- Y is Y ⇒ X

- AppRules = (App nil L L), (App (cons h T), L, (cons h A)) :- App(T,L,A)
- AppRules, (App ’(1 2), ’(3 4), Z) → Z=’(1 2 3 4)
- AppRules, (App ’(1 2), Y, ’(1 2 3 4)) → Y=’(3 4)
- AppRules, (App X, Y, ’(1 2 3 4)) → X=nil, Y=’(1 2 3 4), … (more solutions)

- An example of *declarative* programming
- Prolog searches in a way that may lead to looping, provides support to control search, etc.

Connections with ACL2

For any FO ϕ, we can find a universal ψ in an *expanded* language such that ϕ is satisfiable iff ψ is satisfiable.

$\langle \forall u, v \ \langle \exists z \ \phi(u, v, z) \rangle \rangle$ $\langle \forall u, v \ \langle \exists z \ (App \ u \ v) = (Rev \ z) \rangle \rangle$

First, PNF, and push existentials left (2nd order logic)

$\langle \exists F_z \ \langle \forall u, v \ \phi(u, v, F_z(u, v)) \rangle \rangle$ $\langle \exists F_z \ \langle \forall u, v \ (App \ u \ v) = (Rev \ (F_z \ u \ v)) \rangle \rangle$

Previously, we saw how to go back to FO while preserving SAT with

$\langle \forall u, v \ \phi(u, v, F_z(u, v)) \rangle$ $\langle \forall u, v \ (App \ u \ v) = (Rev \ (F_z \ u \ v)) \rangle$

But what about preserving validity? This method doesn't work, as we've seen. Can we make it work in a FO setting?

This is how ACL2 handles quantifiers

DEMO

$\langle \forall u, v \ \langle \exists z \ (App \ u \ v) = (Rev \ z) \rangle \rangle$

$\longrightarrow$

$\langle \forall u, v \ (E_z \ u \ v) \rangle$

As above, but not enough

$(E_z \ u \ v) \ \equiv \ (App \ u \ v) = (Rev \ (F_z \ u \ v))$

Constrain F_z:

$(App \ u \ v) = (Rev \ z) \ \Rightarrow \ (E_z \ u \ v)$

if (App *u v*) = (Rev *z*) has solution

then F_z is also a solution

Dealing with Equality

- Plan for a FO validity checker w/=: Given FO ϕ, negate & Skolemize to get universal ψ s.t. Valid(ϕ) iff Unsat(ψ). Convert ψ into equivalent CNF $\mathcal{K}$. Generate ψ^* in expanded language wout/= s.t. Sat(ψ) iff Sat(ψ^*). Use U-Resolution: Unsat(ψ^*) iff $\varnothing \in URes_\omega(\mathcal{K})$ iff $\exists n$ s.t. $\varnothing \in URes_n(\mathcal{K})$
- To go from ψ to ψ^*
  - Introduce a new binary relation symbol, E
  - Replace $t_1=t_2$ *with* $E(t_1, t_2)$ everywhere in ψ
  - Force E to be an equivalence relation by adding clauses
    - $\{E(x,x)\}$, $\{\neg E(x,y), E(y,x)\}$, $\{\neg E(x,y), \neg E(y,z), E(x,z)\}$
  - Force E to be a congruence
    - $\{\neg E(x_1,y_1),\ldots,\neg E(x_n,y_n), E(f(x_1,\ldots,x_n), f(y_1,\ldots,y_n))\}$ for every n-ary f in ψ
    - $\{\neg E(x_1,y_1),\ldots,\neg E(x_n,y_n), \neg R(x_1,\ldots,x_n), R(y_1,,,,y_n)\}$ for every n-ary R in ψ
  - Notice all the clauses are Horn!