Logical Theories

Revisit the Logic

- Recall the logic we use the following logic:
  - Goals: \( G ::= L \mid true \mid G_1 \land G_2 \mid H \Rightarrow G \mid \forall x. G \)
  - Hypotheses: \( H ::= L \mid true \mid H_1 \land H_2 \)
  - Literals: \( L ::= p(E_1, \ldots, E_n) \)
  - Expressions: \( E ::= n \mid f(E_1, \ldots, E_n) \)
- This is a subset of FOL
  - Formulas such as \( \forall x. p \), \( (\forall x. p) \Rightarrow Q \) are not (yet) allowed
  - This is sufficient for VCGen if:
    - The invariants, preconditions and postcond. are all from \( H \)

A Semantic for Our Logic

- Define validity (truth of \( G \)):
  - Each predicate symbol has a meaning: \( p :: \mathbb{Z} \rightarrow \mathbb{B} \)
  - Each expression symbol has a meaning: \( f :: \mathbb{Z} \rightarrow \mathbb{B} \)
- We give meaning to each formula:
  - \( G \) means that the (closed) formula \( G \) holds
  - \( G \models true \)
  - \( G \models G_1 \land G_2 \) when \( G_1 \models G_2 \)
  - \( G \models \forall x. G \) when for all \( n \in \mathbb{Z} \), we have \( G(n/x) \)
  - \( H \models G \) when \( H \models G \)
  - \( G \models p(E_1, \ldots, E_n) \) when \( p [[E_1]], \ldots, [E_n] ] = true \)

The Theorem Proving Problem

- Write an algorithm "prove" such that:
  - If \( \text{prove}(G) = true \) then \( G \)
    - Soundness, most important
  - If \( G \models true \) then \( \text{prove}(G) = true \)
    - Completeness, nice to have

Recall: Derivation Rules for Assertions

- We define a judgment \( \vdash A \) inductively:
  \[
  \begin{align*}
  \vdash A & \quad \vdash B \\
  \vdash A \land B & \quad \vdash \forall x. A \\
  \vdash [a/x]A & \quad \vdash [E/x]A \\
  \vdash A & \quad \vdash A \Rightarrow B \\
  \vdash B & \quad \vdash \forall x. A \Rightarrow [E/x]A \\
  \vdash [a/x]A & \quad \vdash [E/x]A \\
  \vdash A & \quad \vdash B \\
  \vdash B & \quad \vdash \exists x. A \Rightarrow B
  \end{align*}
  \]
A Theorem Prover for our Logic

- We must work symbolically
  - Or otherwise how can we hope to check \( m \in \mathbb{Z} \Rightarrow G[n/x] \)?
- Define the following symbolic "prove" algorithm
  - \( \text{prove}(H, G) \) - prove the goal \( H \Rightarrow G \)
  - \( \text{prove}(H, \text{true}) = \text{true} \)
  - \( \text{prove}(H, G \land G') = \text{prove}(H, G) \land \text{prove}(H, G') \)
  - \( \text{prove}(H, H \Rightarrow G) = \text{prove}(H \land H, G) \)
  - \( \text{prove}(H, \forall x, G) = \text{prove}(H, G[a/x]) \) (\( a \) is "fresh")
  - \( \text{prove}(H, L) = \text{Unsat}(H \land \neg L) \)

Soundness

- Assume for now that \text{Unsat} is sound and complete
  - \( \text{Unsat}(H_1 \land \ldots \land H_n \land \neg L_0) \iff \forall a_1 \ldots a_n . H_1 \land \ldots \land H_n \Rightarrow L_0 \)
- Easy to show "prove" is sound
- Can also show "prove" is complete
- No search really
- Goal-directed procedure, very efficient
- Why? Because we use FOL only superficially ...

How Powerful is Our Prover?

- With \text{VCGen} in mind we must restrict invariants to
  \( H_1 \land \ldots \land H_n \land \neg L_0 \)
- No disjunction, implication or quantification!
  - Just sets of literals. Is that too little?
- Consider the function:
  ```c
  void insert(LIST *a, LIST * b) {
    LIST ** t = a->next; a->next = b; b->next = t;
  }
  ```
- And the problem is to verify that
  - It preserves linearity: all list cells are pointed to by at most
    one other list cell
  - Provided that \( b \) is non-NULL and not pointed to by any cell

A Theorem Prover for Literals

- We have reduced the problem to:
  - \( \text{prove}(H, L) \)
- But \( H \) is a conjunction of literals
- Thus we have to prove that \( L_1 \land \ldots \land L_n \Rightarrow L \)
- Or equivalently, that \( L_1 \land \ldots \land L_n \land \neg L \) is unsatisfiable
  - For any assignment of values to parameters \( a_j \), the truth value of
    the conjunction of literals is false
- Now we can say that
  - \( \text{prove}(H, L) = \text{Unsat}(H \land \neg L) \)

VCGen and Prove

- \text{VCGen} uses connectives:
  - \( \land \) to construct sets of proof obligations
  - \( \forall \) to model use of "fresh" variables
  - \( \Rightarrow \) to model assumptions
- \text{Prove} handles connectives:
  - \( \land \) : prove conjuncts in turn
  - \( \forall \) : introduce a "fresh" parameter
  - \( \Rightarrow \) : collect assumptions

Lists and Linearity

- A bit of formal notation (remember the sel/upd):
  - We write \( \text{sel}(n, a) \) to denote the value of "a-next" given the
    state of the "next" field is "n"
  - We write \( \text{upd}(n, a, b) \) to denote the new state of the "next"
    field after "a-next = b"
  - Code is : \text{void insert(LIST *a, LIST * b) { \text{LIST ** t = a->next; a->next = b; b->next = t; \}}}
  - \text{Pre} is \( (q, q = 0 \Rightarrow \forall p_1/p_2 . \text{sel}(n, p_2) = \text{sel}(n, p_2) \Rightarrow q = q \land q = q) \)
  - \text{Post is} \( (q, q = 0 \Rightarrow \forall p_1/p_2 . \text{sel}(n, p_2) = \text{sel}(n, p_2) \Rightarrow q = q \land q = q) \)
  - \text{VC is} \( \text{Pre} \Rightarrow \text{Post}(\text{upd}(n, a, b), \text{sel}(n, a)) / n \) \text{ Not a G} !
Two Solutions

- So it is quite easy to want to step outside H
- We can do two things:
  1. Extend the language of H
     - And then extend the prover
  2. Push the complexity of invariants into literals
     - And then extend the unsatisfiability procedure

Goal Directed Theorem Proving (1)

- Say that we extend the use of quantifiers:
  \[ G \equiv L \land \forall x \cdot G \Rightarrow G \lor \exists x \cdot G \]
  \[ H \equiv L \land \forall x \cdot H \lor \exists x \cdot H \]
- We have also introduced an existential choice
  - Both in "H = \exists x \cdot G" and "\forall x \cdot H = G"
- Existential choices are postponed
  - Introduce unification variables + unification
    \[ \text{prove}(H, G, x) = \text{prove}(H, G[u/x]) \quad (u \text{ is a unif var}) \]
  - Still sound and complete goal directed proof search !
    - Provided that Unsat can handle unification variables !

Goal Directed Theorem Proving (2)

- We can add disjunction (to goals):
  \[ G \equiv \text{true} \land G_1 \land G_2 \land H \Rightarrow G \land \forall x \cdot G_1 \lor G_2 \]
- Extend prover as follows:
  \[ \text{prove}(H, G_1 \lor G_2) = \text{prove}(H, G_1) \lor \text{prove}(H, G_2) \]
- This introduces a choice point in proof search
  - Called a "disjunctive choice"
  - Backtracking is complete for this choice selection
  - But only in intuitionistic logic !

Goal Directed Theorem Proving (3)

- Now we extend a bit the language of hypotheses
  - Important since this adds flexibility for invariants and specs.
  \[ H \equiv L \land \text{true} \land \forall x \cdot H \land G \Rightarrow H \]
- We extend the prover as follows:
  \[ \text{prove}(H, G, \forall x \cdot H) \Rightarrow G) \equiv \]
  \[ \text{prove}(H, G) \lor (\text{prove}(H \land H, G) \land \text{prove}(H, G)) \]
  - This adds another choice (clause choice in Prolog) expressed here also as a disjunctive choice
  - Still complete with backtracking

Goal Directed Theorem Proving (4)

- The VC for linear lists can be proved in this logic !
  - This logic is called Hereditary Harrop Formulas
  - But the prover is not complete in a classical sense
    - And thus complications might arise with certain theories
  - Still no way to have disjunctive hypotheses
    - The prover becomes incomplete even in intuitionistic logic
      - E.g., cannot prove even that \( P \lor Q = Q \lor P \)
  - Let’s try the other method instead ...

Constructive vs. Classical Proofs

- Classical logic has the axiom \( P \lor \neg P \)
  - Cannot prove this fact in intuitionistic logic
  - We can prove fewer facts in intuitionistic logic

- Puzzle: Prove the following fact:
  \[ \exists x \in R \cdot x^2 \in Q \]
- Hint: Try \( \sqrt{2^2} \)
A Theory of Linear Lists

- Push the complexity into literals
- Define new literals:
  \[ \text{linear}(n) \equiv q \cdot \forall n_p. \exists(n, n) = q \rightarrow p \]
  \[ \text{rcO}(n, b) \equiv \forall b, q = 0 \rightarrow \exists(n, p) = b \]
- Now the predicates become:
  
  \[ \text{Pre} \equiv \text{linear}(n) \land \text{rcO}(n, b) \land a = 0 \land b = 0 \]
  \[ \text{Post} \equiv \text{linear}(n) \land \text{rcO}(n, b) \land a = 0 \land b = 0 \rightarrow \text{linear}(\text{update}(a, b, n, a))) \]

  This is a G.
- The hard work is now in the satisfiability procedure

Discussion

- It makes sense to push hard work in literals:
  - Can be handled in a customized way within the Sat procedures
  - The hand-crafted inference rules guide the proof
  - The inference rules are useful lemmas
  - An important technique
- Just like in type inference, or data flow analysis:

<table>
<thead>
<tr>
<th>Theorem Proving</th>
<th>Type Inference</th>
<th>Data Flow Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Literals</td>
<td>Type system</td>
<td>Lattice</td>
</tr>
<tr>
<td>Derivation rules</td>
<td>Typing rules</td>
<td>Transfer functions</td>
</tr>
<tr>
<td>Sat. procedure</td>
<td>Inference algorithm</td>
<td>Iterative D.F.A.</td>
</tr>
</tbody>
</table>

Examples of Theories

- Symbols: 0, 1, -1, 2, -2, ..., +, *, \(=\), \(+\), \((-)\) (with the usual meaning)
  - Theory of integers with arithmetic (Presburger arithmetic)

- Theory of total orders:
  - Symbols: \(\forall, \leq\)
  - Axioms: transitivity, anti-symmetry and \(\forall x \forall y. x \leq y \land y \leq x \rightarrow x = y\)

- Theory of lists...

Theories

- Now we turn to \(\text{unsat}(L_1, \ldots, L_n)\)

- A theory consists of:
  - A set of function and predicate symbols (syntax)
  - Definitions for the meaning of these symbols (semantics)
  - Semantic or axiomatic definitions

Decision Procedures for Theories

- The Decision Problem:
  - Decide whether a formula in a theory + FOL is true

  Example:
  - Decide whether \(\forall x. x > 0 = (\exists y. x \leq y + 1) \in (\{+, \cdot, =\})\)

- A theory is decidable when there is an algorithm that solves the decision problem for the theory
  - This algorithm is the decision procedure for the theory
Satisfiability Procedures for Theories

- The Satisfiability Problem
  - Decide whether a conjunction of literals in the theory is satisfiable
  - Factor out the FOL part of the decision problem
  - The decision problem can be reduced to the satisfiability problem
    - parameters for \( \land \), skolem functions for \( \exists \), negate and convert to DNF
- This is what we need to solve in our simple prover
- We will explore a few useful theories and satisfiability procedures for them...

Examples of Satisfiability Problems

- Linear arithmetic
  - Symbols: \( \geq, =, +, \cdot \), integer literals
  - Example: \( y \cdot 2x + 1, y + x \geq 1, y > 0 \) is unsat
  - Satisfiability problem is in \( \Pi \)
- Lists
  - Symbols: cons, car, cdr, atom, nil
    
    \[
    \begin{array}{ll}
    \text{atom(nil)} & \text{car(cons(x,y))} = x \\
    \text{cdr(cons(x,y))} & = y
    \end{array}
    \]
  - Theorem: \( \text{car(x)} = \text{car(y)} \land \text{cdr(x)} = \text{cdr(y)} = x = y \)
- Arrays
  - Theorem: \( \text{upd}(x, y, \text{sel}(x, y)) = x \)

Examples of Theories. Equality.

- The theory of equality with uninterpreted functions
  - Symbols: \( =, f, g, \ldots \)
  - Axiomatically defined:
    \[
    \begin{align*}
    E & = E \\
    E_1 = E_2 & \iff E_1 = E_2 \\
    E_1 = E_2 & \iff E_1 = E_2 \\
    f(E_1) & = f(E_2)
    \end{align*}
    \]
  - Example of a satisfiability problem:
    \[
    g(g(g(x))) = x \land g(g(g(g(g(x))))) = x \land g(x) = x
    \]

Mixed Theories

- Often we have facts involving symbols from multiple theories
- Example:
  - \( E \)'s symbols: \( =, f, g, \ldots \) (uninterpreted functions)
  - \( R \)'s symbols: \( =, +, \cdot, \leq, 0, 1, \ldots \) (linear arithmetic)
  - Fact: \( f(f(x) - f(y)) = f(z), x \leq y, y \cdot z \leq x, z \geq 0 \)
- We may have sat procedures for each theory
  - \( E \)'s sat procedure by Ackerman in 1924, \( R \)'s proc. by Fourier
  - The sat procedure for combination is much harder
    - Only in 1979 we got \( E \cdot R \)

Satisfiability of Mixed Theories

- Again: \( F = f(f(x) - f(y)) = f(z), x \leq y, y \cdot z \leq x, z \geq 0 \)
- Separate \( F \) = \( F_1 \land F_2 \) such that
  - \( F \) is sat \iff \( F_1 \) \land \( F_2 \) is sat
    - Note: equi-satisfiable is not the same as equivalence
  - \( F_1 \) (from \( E \)) \iff \( g_1 = f(x), g_2 = f(y), f(g_3) = f(z) \)
  - \( F_2 \) (from \( R \)) \iff \( x \leq y, y \cdot z \leq x, z \geq 0, g_3 = g_1 = g_2 \)
- Both \( F_1 \) and \( F_2 \) are independently satisfiable
  - But their conjunction is not

Idea for Satisfiability of Mixed Theories

- The problem with independent satisfiability is that the satisfying assignments may be incompatible
- Idea (Nelson and Oppen): Each satisfiability procedure should also announce all equalities between variables that it discovers
  - \( F_1 \) (from \( E \)) \iff \( g_1 = f(x), g_2 = f(y), f(g_3) = f(z) \)
  - \( F_2 \) (from \( R \)) \iff \( x \leq y, y \cdot z \leq x, z \geq 0, g_3 = g_1 = g_2 \)
  - \( F_1 \) announces \( x = y \)
  - \( F_2 \) announces \( g_3 = g_1 \)
  - \( F_1 \) announces \( g_1 \)
  - \( F_2 \) discovers unsatisfiability
When Does This Not Work?

**Counterexample 1:**
- Theory 1: \( \exists a, b \, \forall x \, x = a \lor x = b \)
- Theory 2: \( \exists a, b, c \, a = b \land b = c \land a = c \)
- Empty conjunction is unsat, but Nelson-Oppen cannot prove it

**Counterexample 2:**
- \( E \): integer linear arithmetic
- Facts: int(x), \( 1 \leq x, x \leq 2 \), \( a = 1, b = 2 \), \( f(x) = f(a), f(x) = f(b) \)
- int(x) is an infinite disjunction!