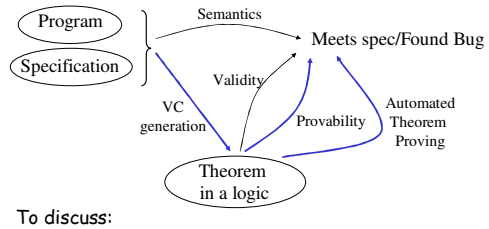
Logical Theories

Where Are We?



- To discuss:
  - Validity of VCs
  - Provability of VCs
  - Automation of provability (automated theorem proving)

Revisit the Logic

- Recall the we use the following logic:

Goals: $G ::= L \mid true \mid G_1 \wedge G_2 \mid H \Rightarrow G \mid \forall x.\, G$

Hypotheses: $H ::= L \mid true \mid H_1 \wedge H_2$

Literals: $L ::= p(E_1, \ldots, E_k)$

Expressions: $E ::= n \mid f(E_1, \ldots, E_m)$

- This is a subset of FOL
  - Formulas such as "$P_1 \vee P_2$", "$(\forall x.\, P) \Rightarrow Q$" 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 VC)
  - Each predicate symbol has a meaning: $[\![p]\!] : \mathbb{Z}^k \to \mathbb{B}$
  - Each expression symbol has a meaning: $[\![f]\!] : \mathbb{Z}^n \to \mathbb{Z}$
- We give meaning to each formula:

  $\models G$ means that the (closed) formula G holds

$\models true$

$\models G_1 \wedge G_2$ when $\models G_1$ and $\models G_2$

$\models \forall x.G$ when for all $n \in \mathbb{Z}$ we have $\models G[n/x]$

$\models H \Rightarrow G$ when "$\models G$ whenever $\models H$"

$\models p(E_1, \ldots, E_k)$ when $[\![p]\!]([\![E_1]\!], \ldots, [\![E_n]\!]) = true$

The Theorem Proving Problem

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

Recall: Derivation Rules for Assertions

- We define a judgment $\vdash A$ inductively:

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

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

A Theorem Prover for our Logic

- We must work symbolically
  - Or otherwise how can we hope to check $\forall n \in \mathbb{Z} \models G[n/x]$?
- Define the following symbolic "prove" algorithm
  - prove(H, G) - prove the goal "$H \Rightarrow G$"

prove(H, true) = true
prove(H, $G_1 \wedge G_2$) = prove(H, G_1) && prove(H, G_2)
prove(H, $H_1 \Rightarrow G_2$) = prove($H \wedge H_1$, G_2)
prove(H, $\forall x.\ G$) = prove(H, $G[a/x]$) (a is "fresh")
prove(H, L) = ?

A Theorem Prover for Literals

- We have reduced the problem to:
  prove(H, L)
- But H is a conjunction of literals
- Thus we have to prove that $L_1 \wedge \ldots \wedge L_k \Rightarrow L$
- Or equivalently, that $L_1 \wedge \ldots \wedge L_k \wedge \neg L$ is <u>unsatisfiable</u>
  - For any assignment of values to parameters a_j the truth value of the conjunction of literals is false
- Now we can say that
  prove(H, L) = Unsat($H \wedge \neg L$)

Soundness

- Assume for now that Unsat is sound and complete
  - Unsat($H_1 \wedge \ldots \wedge H_{k-1} \wedge \neg L_k$) iff $\models \forall a_1 \ldots H_1 \wedge \ldots \wedge H_{k-1} \Rightarrow L_k$
- 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 …

VCGen and Prove

- VCGen uses connectives:
  - $\wedge$ to construct sets of proof obligations
  - $\forall$ to model use of "fresh" variables
  - $\Rightarrow$ to model assumptions
- Prove handles connectives:
  - $\wedge$: prove conjuncts in turn
  - $\forall$: introduce a "fresh" parameter
  - $\Rightarrow$: collect assumptions

How Powerful is Our Prover?

- With VCGen in mind we must restrict invariants to
  $H ::= L \mid true \mid H_1 \wedge H_2$
- No disjunction, implication or quantification !
  - Just sets of literals. Is that too little ?
- Consider the function:

```
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 <u>linearity</u>: 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

Lists and Linearity

- A bit of formal notation (remember the sel/upd):
  - We write sel(n, a) to denote the value of "a->next" given the state of the "next" field is "n"
  - We write upd(n, a, b) to denote the new state of the "next" field after "a->next = b"

Code is `void insert(LIST *a, LIST * b) { LIST *t = a->next; a->next = b; b->next = t; }`

Pre is $(\forall q.\ q \neq 0 \Rightarrow \forall p_1.\forall p_2.\ sel(n, p_1) = sel(n, p_2) = q \Rightarrow p_1 = p_2)$
$\wedge\ b \neq 0 \wedge \forall p.\ sel(n, p) \neq b) \wedge a \neq 0$

Post is $(\forall q.\ q \neq 0 \Rightarrow \forall p_1.\forall p_2.\ sel(n, p_1) = sel(n, p_2) = q \Rightarrow p_1 = p_2)$

VC is Pre $\Rightarrow$ Post[upd(upd(n, a, b), b, sel(n, a)) / n] 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:

$G ::= L \mid true \mid G_1 \wedge G_2 \mid H \Rightarrow G \mid \forall x.\, G \mid \underline{\exists x.\, G}$

$H ::= L \mid true \mid H_1 \wedge H_2 \mid \underline{\forall x.\, H}$

- We have also introduced an existential choice
  - Both in "$H \Rightarrow \exists x.\, G$" and "$\forall x. H \Rightarrow G$"
- Existential choices are <u>postponed</u>
  - Introduce unification variables + unification
    - prove(H, $\exists x.G$) = prove(H, $G[u/x]$) (u is a unif var)
    - prove(H, u = t) = instantiate u with t if $u \notin FV(t)$
- 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):

$G ::= true \mid L \mid G_1 \wedge G_2 \mid H \Rightarrow G \mid \forall x.\, G \mid \underline{G_1 \vee G_2}$

- Extend prover as follows:

  prove(H, $G_1 \vee G_2$) = prove(H, G_1) || 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 !

Constructive vs. Classical Proofs

- Classical logic has the axiom $P \vee \neg P$
  - Cannot prove this fact in intuitionistic logic
  - We can prove fewer facts in intuitionistic logic
- Puzzle: Prove the following fact:

$$\exists x \in R \backslash Q.\ \ x^{\sqrt{2}} \in Q$$

- Hint: Try $\sqrt{2}^{\sqrt{2}}$

Goal Directed Theorem Proving (3)

- Now we extend a bit the language of hypotheses
  - Important since this adds flexibility for invariants and specs.

$$H ::= L \mid true \mid H_1 \wedge H_2 \mid \underline{G \Rightarrow H}$$

- We extend the prover as follows:

prove(H, $(G_1 \Rightarrow H_1) \Rightarrow G$) =

  prove(H, G) || (prove($H \wedge H_1$, G) && prove(H, G_1))

  - 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 $P \vee Q \Rightarrow Q \vee P$
- Let's try the other method instead …

A Theory of Linear Lists

- Push the complexity into literals
- Define new literals:

  $linear(n) =_{def} \forall q.\ q \neq 0 \Rightarrow \forall p_1.\forall p_2.\ sel(n, p_1) = sel(n, p_2) = q \Rightarrow p_1 = p_2$

  $rc0(n, b) =_{def} b \neq 0 \Rightarrow \forall p.\ sel(n, p) \neq b$

- Now the predicates become:

Pre is $linear(n) \wedge rc0(n, b) \wedge a \neq 0 \wedge b \neq 0$

Post is $linear(n)$

VC is $linear(n) \wedge rc0(n, b) \wedge a \neq 0 \wedge b \neq 0 \Rightarrow$
$linear(upd(upd(n, a, b), b, sel(n, a))))$ This is a *G* !

- The hard work is now in the satisfiability procedure

A Theory of Linear Lists

- In order to allow the prover to work with "linear" and "rc0" we must define their meaning:
  - Semantically (by giving the definitions from before)
  - Axiomatically (by giving a set of axioms that define them):

$$\frac{linear(n) \quad a \neq 0 \quad rc0(n, b)}{linear(upd(n, a, b))} \qquad \frac{linear(n) \quad a \neq 0 \quad rc0(n, b)}{rc0(upd(n, a, b), sel(n, a))}$$

- Now we can prove the VC with just three uses of these axioms
- Is this set of axioms sound and complete?

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 prover
  - The inference rules are useful lemmas
  - An important technique
- Just like in type inference, or data flow analysis :

Theorem Proving	Type Inference	Data Flow Analysis
Literals	Type system	Lattice
Derivation rules	Typing rules	Transfer functions
Sat. procedure	Inference algorithm	Iterative D.F.A.

Theories

- Now we turn to unsat(L_1, ..., L_k)
- A <u>theory</u> consists of:
  - A set of function and predicate symbols (syntax)
  - Definitions for the meaning of these symbols (semantics)
    - Semantic or axiomatic definitions

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: { =, ≤ }
  - Axioms: transitivity, anti-symmetry and $\forall x \forall y.\ x \leq y \vee y \leq x$
- Theory of lists ...

Decision Procedures for Theories

- The <u>Decision Problem</u>:
  - Decide whether a formula in a theory + FOL is true
- Example:
  - Decide whether $\forall x.\ x > 0 \Rightarrow (\exists y.\ x = y + 1)$ in $\{\mathbb{N}, +, =, >\}$
- A theory is decidable when there is an algorithm that solves the decision problem for the theory
  - This algorithm is the <u>decision procedure</u> 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 $\forall$, 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 Theories. Equality.

- The theory of equality with uninterpreted functions
- Symbols: $=, \neq, f, g, \ldots$
- Axiomatically defined:

$$\frac{}{E = E} \qquad \frac{E_2 = E_1}{E_1 = E_2} \qquad \frac{E_1 = E_2 \quad E_2 = E_3}{E_1 = E_3} \qquad \frac{E_1 = E_2}{f(E_1) = f(E_2)}$$

- Example of a satisfiability problem:

$$g(g(g(x)) = x \wedge g(g(g(g(g(x))))) = x \wedge g(x) \neq x$$

Examples of Satisfiability Problems

- Linear arithmetic
  - Symbols: $\geq, =, +, -$, integer literals
  - Example: $y > 2x + 1, y + x > 1, y < 0$ is unsat
  - satisfiability problem is in P
- Lists
  - Symbols: cons, car, cdr, atom, nil

$$\frac{}{atom(nil)} \qquad \frac{}{car(cons(x,y)) = x} \qquad \frac{}{cdr(cons(x,y)) = y}$$

  - Theorem: $car(x) = car(y) \wedge cdr(x) = cdr(y) \Rightarrow x = y$
- Arrays
  - Theorem: $upd(x, y, sel(x, y)) = x$

Mixed Theories

- Often we have facts involving symbols from multiple theories
- Example:
  - E's symbols $=, \neq, f, g, \ldots$ (uninterpreted functions)
  - R's symbols $=, \neq, +, -, \leq, 0, 1, \ldots$ (linear arithmetic)
  - Fact: $f(f(x) - f(y)) \neq f(z), x \leq y, y + z \leq x, z \geq 0$
- 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+R

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
  - Note: equi-satisfiable is not the same as equivalence
  - 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

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_E (from E) = $g_1 = f(x), g_2 = f(y), f(g_3) = f(z)$
  - F_R (from R) = $x \leq y, y + z \leq x, z \geq 0, g_3 = g_1 - g_2$
  - F_R announces $x = y$
  - F_E announces $g_1 = g_2$
  - F_R announces $g_3 = z$
  - F_E discovers unsatisfiability

When Does This Not Work?

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