CS 3110

Lecture 22: Mechanized Logic

Prof. Clarkson

Fall 2014

Today's music: "Mr. Roboto" by Styx

The problem's plain to see: too much technology.

Machines to save our lives. Machines dehumanize.

Review

Current topic:

- How to reason about correctness of code
- Started with informal arguments
- Developed formal logic

Today:

- A proof assistant called Coq

Question #1

How much of PS5 have you finished?

A. None
B. About 25%
C. About 50%
D. About 75%
E. I'm done!!!

Review: Proof rules of IPC, part 1

<table>
<tr><th>Rule name</th><th>Rule</th></tr>
<tr><td>/\ intro</td><td>if F |- f1 and F |- f2 then F |- f1 /\ f2</td></tr>
<tr><td>/\ elim L</td><td>if F |- f1 /\ f2 then F |- f1</td></tr>
<tr><td>/\ elim R</td><td>if F |- f1 /\ f2 then F |- f2</td></tr>
<tr><td>=> elim</td><td>if F |- f and F |- f => g then F |- g</td></tr>
<tr><td>=> intro</td><td>if F, f |- g then F |- f => g</td></tr>
<tr><td>assump</td><td>f |- f</td></tr>
<tr><td>weak</td><td>if F |- f then F,g |- f</td></tr>
<tr><td>set assump</td><td>F,f |- f</td></tr>
</table>

Review: Proof rules of IPC, part 2

<table>
<tr><th>Rule name</th><th>Rule</th></tr>
<tr><td>\/ intro L</td><td>if F |- f1 then F |- f1 \/ f2</td></tr>
<tr><td>\/ intro R</td><td>if F |- f2 then F |- f1 \/ f2</td></tr>
<tr><td>\/ elim</td><td>if F |- f1 \/ f2 and F |- f1 => g and F |- f2 => g then F |- g</td></tr>
<tr><td>true intro</td><td>F |- true</td></tr>
<tr><td>false elim</td><td>if F |- false then F |- f</td></tr>
<tr><td>~ intro</td><td>if F |- f => false then F |- ~f</td></tr>
<tr><td>~ elim</td><td>if F |- ~f then F |- f => false</td></tr>
</table>

Review: Proof rules of IQC

Rule name	Rule
---	All rules of IPC
forall intro	if `F |- f(x)` and `x` not in FV(`F`) then `F |- forall x, f(x)`
forall elim	if `F |- forall x, f(x)` then `F |- f(t)`
exists intro	if `F |- f(t)` then `F |- exists x, f(x)`
exists elim	if `F |- exists x, f(x)` and `F |- f(x) => g` and `x` not in FV(`F,g`) then `F |- g`

Theories

- IQC reaches its full power when augmented with *theories*
- Collections of
  - names of relations and functions, and
  - new proof rules for those

Theory of equality

- Relation: `equals(t1,t2)`
  - normally written `t1=t2`
- Proof rules:
  - reflexivity: `t=t`
  - symmetry: if `t1=t2` then `t2=t1`
  - transitivity: if `t1=t2` and `t2=t3` then `t1=t3`
  - eq-fn: if `t1=u1` and...and `tn=un` then `fn(t1,...,tn) = fn(u1,...,un)`
  - eq-rel: if `t1=u1` and...and `tn=un` then `R(t1,...,tn) = R(u1,...,un)`

Theory of rings

- *Ring*: mathematical structure that abstracts addition and multiplication
  - see Math 4320
- Relies on theory of equality
- Functions:
  - **`plus(t1, t2)`** and **`mult(t1,t2)`** and **`neg(t)`**
    - written **`t1+t2`** and **`t1*t2`** and **`-t`**
  - **`zero`** and **`one`**
    - written **`0`** and **`1`**

Theory of rings

- Proof rules (all are axioms):
  - `forall a b c, (a+b)+c = a+(b+c)`
  - `forall a b, a+b = b+a`
  - `forall a, 0+a = a`
  - `forall a, a + (-a) = 0`
  - `forall a b c, a*(b+c) = (a*b)+(a*c)`
  - `forall a b c, (b+c)*a = (b*a)+(c*a)`
  - `forall a b c, (a*b)*c = a*(b*c)`
  - `forall a b, a*b = b*a`
  - `forall a, 1*a = a`
- Syntactic sugar:
  `forall a b, f`
  means `forall a, (forall b, f)`

Prelim 2

- One week from today
- Covers everything from Oct 2 through Nov 12 (inclusive)
  - People with Thursday recitations, note that today's recitation is included
- Sample prelim posted on Piazza
- Review session in recitation day before prelim
- Cancel lecture on day of prelim
- You can take prelim at your choice of 5:30-7:00 pm or 7:30-9:00 pm; no need to reserve in advance
- Three rooms, will be assigned by netid next week
- Closed book
  - But you may have one page of notes
  - 8.5x11" two-sided ☺

Why formal logic?

- Humans make mistakes in writing proofs
- Humans make mistakes in checking proofs
- Formal logic:
  - Reduces proof to symbolic manipulation
  - Maybe a machine could check that manipulation
- Analogy:
  - Compiler type checks program
  - Proof checker uses proof rules we've given to check proof

Mechanized proof

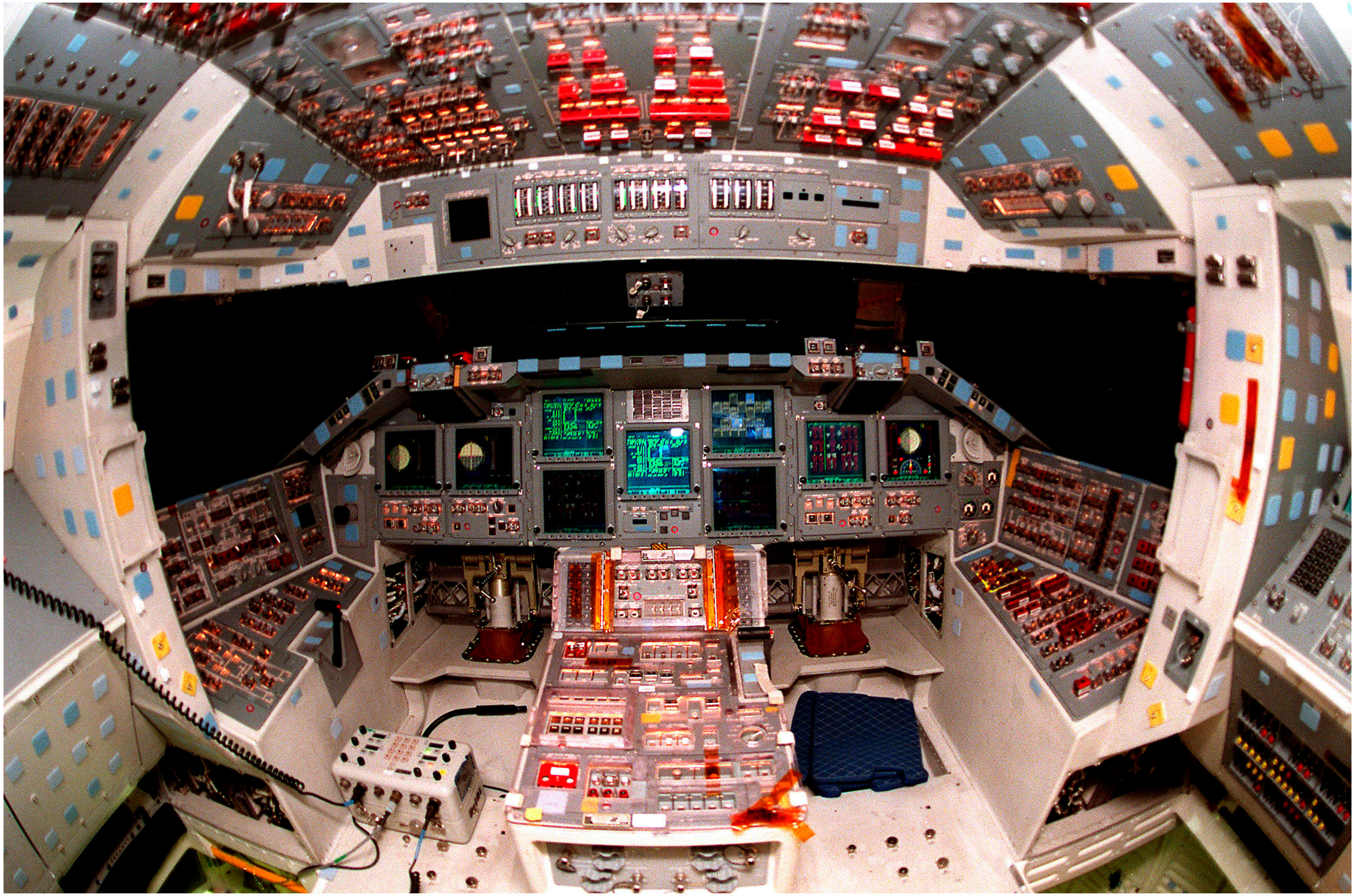
- Automated theorem provers
  - You give tool a theorem
  - Tools finds a proof or a counterexample
    - Or runs out of time
  - e.g., Z3, developed at Microsoft
    - Ships with the Windows 7 device driver developer's kit
- Proof assistants
  - You give tool a theorem
  - You and tool cooperatively find proof
    - Human guides the construction
    - Machine does the low-level details
  - e.g., Coq, Isabelle/HOL, NuPRL
    - NuPRL: Prof. Constable (Cornell)
    - Coq: used to verify compiler, OS kernel, etc.

Coq

- **1984:** Coquand and Huet first begin implementing a new theorem prover Coq based on *calculus of inductive constructions*
- **1992:** Coq ported to Caml
- **2012:** Coq version 8.4
  - Implemented in OCaml
  - Can produce verified OCaml code

Thiery Coquand
1961 –

Coq's full system



Subset of Coq we'll use

Coq3110.v

- We went through the file up through and including implication and forall.

Please hold still for 1 more minute

WRAP-UP FOR TODAY

Upcoming events

- **PS5 due tonight**
- Prelim 2 in one week

This is mechanized.

THIS IS 3110