CS 671 Automated Reasoning

Tactical Theorem Proving in NuPRL

1. **Basic Tactics**
2. **Tacticals**
3. **Advanced Tactics**
   Chaining, Induction, Case Analysis

Tactics: User-defined inference rules

- **Meta-level programs built using**
  - Basic inference rules
  - Predefined tacticals …
  - Meta-level analysis of the proof goal and its context
  - Large collection of standard tactics in the library
- **May produce incomplete proofs**

  ↦ User has to complete the proof by calling ither tactics
- **May not terminate**

  ↦ User has to interrupt execution

but

Applying a tactic always results in a valid proof

Basic Tactics

Subsume primitive inferences under a common name

- `Hypothesis`: *Prove* $\ldots C \ldots \vdash C'$ *where* C' *α-equal to* C
  `Declaration`: *Prove* $\ldots x{:}T \ldots \vdash x \in T'$ *where* T' *α-equal to* T
  - Variants: `NthHyp` i, `NthDecl` i
- `D` c: *Decompose the outermost connective of clause c*
- `EqD` c: *Decompose immediate subterms of an equality in clause c*
  `MemD` c: *Decompose subterm of a membership term in clause c*
  - Variants: `EqCD` , `EqHD` i, `MemCD` , `MemHD` i
- `EqTypeD` c: *Decompose type subterm of an equality in clause c*
  `MemTypeD` c: *Decompose type subterm of a membership term in clause c*
  - Variants: `EqTypeCD` , `EqTypeHD` i, `MemTypeCD` , `MemTypeHD` i
- `Assert` t: *Assert (or cut) term t as last hypothesis*
- `Auto`: *Apply trivial reasoning, decomposition, decision procedures ...*
- `Reduce` c: *Reduce all primitive redices in clause c*

TACTICALS

- tac_1 `THEN` tac_2: *Apply* tac_2 *to all subgoals created by* tac_1

  t `THENL` `[`$tac_1; \ldots; tac_n$`]`: *Apply* tac_i *to the* i*-th subgoal created by* t

  tac_1 `THENA` tac_2: *Apply* tac_2 *to all auxiliary subgoals created by* tac_1

  tac_1 `THENW` tac_2: *Apply* tac_2 *to all wf subgoals created by* tac_1

- tac_1 `ORELSE` tac_2: *Apply* tac_1*. If this fails apply* tac_2 *instead*

- `Try` *tac: Apply tac. If this fails leave the proof unchanged*

- `Complete` *tac: Apply tac only if this completes the proof*

- `Progress` *tac: Apply tac only if that causes the goal to change*

- `Repeat` *tac: Repeat tac until it fails*

  `RepeatFor` i *tac: Repeat tac exactly* i *times*

- `AllHyps` *tac: Try to apply tac to all hypotheses*

  `OnSomHyp` *tac: Apply tac to the first possible hypotheses*

Supplying Parameters to Tactics

- **Position of a hypothesis to be used** — `NthHyp` $\boxed{i}$
- **Names for newly created variables** — `New` $[x]$ `(D 0)`
- **Type of some subterm in the goal** — `With` $\boxed{x : S \rightarrow T}$ `(MemD 0)`
- **Term to instantiate a variable** — `With` $\boxed{s}$ `(D 0)`
- **Universe level of a type** — `At` $\boxed{j}$ `(D 0)`
- **Dependency of a term instance $C[z]$ on a variable z** — `Using` $\boxed{[z,C]}$ `(D 0)`

Advanced Tactics: (Inductive) Analysis

- **Induction**
  - `NatInd` i: standard natural-number induction on hypothesis i
  - `IntInd`, `NSubsetInd`, `ListInd`: induction on $\mathbb{Z}$, $\mathbb{N}$ subranges, lists
  - `CompNatInd` i: complete natural-number induction on hypothesis i
- **Case Analysis**
  - `BoolCases` i: case split over boolean variable in hypothesis i
  - `Cases` [t_1;..;t_n]: $\boldsymbol{n}$-way case split over terms t_i
  - `Decide` P: case split over (decidable) proposition P and its negation

Advanced Tactics: Chaining

- **Instantiating Facts**
  - `InstHyp` [t_1;..;t_n] i: *instantiate hypothesis i with terms $t_1 \ldots t_n$*
  - `InstLemma` *name* [t_1;..;t_n]: *instantiate lemma name with terms $t_1 \ldots t_n$*
- **Forward Chaining**
  - `FHyp` i [h_1;..;h_n]: *forward chain through hypothesis i matching its antecedents against any of the hypotheses $h_1 \ldots h_n$*
  - `FLemma` *name* [h_1;..;h_n]: *forward chain through lemma name*

  Optional argument `Sel` n
- **Backward Chaining**
  - `BHyp` i: *backward chain through hypothesis i matching its consequent against the conclusion of the proof*
  - `BLemma` *name*: *backward chain through lemma name*
  - `Backchain` *bc_names*: *backchain repeatedly through lemmas and hypotheses*

  Optional argument `Using` *binding*

Running Nuprl from a Unix machine

Copy the file `~nuprl/utils/profile/nuprl.config.cs671` to `~/.nuprl.config`

Edit `.nuprl.config` and change the entries

```
(iam "YourNameHere")
(sockets 1289 1980)
```

You may change the 0 to any number between 1-9. DO NOT change 1289!

In an xterm execute

```
xset fp+  nuprl/fonts/bdf
xset fp rehash
xhost +baldwin
rsh baldwin /usr/bin/X11/xterm -display ‘hostname‘:0 -ls
```

Using **baldwin** makes sure that there are no memory issues. You may have to adjust the `-display` setting. You also may want to add`~ nuprl/bin` to your path, e.g. by typing (in csh) `set path = (~ nuprl/bin $path)` into the new window.

On baldwin execute `nuprl/bin/emacsb nuprl`

In emacs type `(m-x)nuprl`

This should run for a minute then pop up the Nuprl windows on the display.

In the navigator, go into the directories `theories`, then `users`, click `MkTHY*`, enter your name into `[token]`, click `OK*` and work only in the newly created theory

To quit, type `stop.` into the emacs shell after the `ML[(ORB)]>` prompt.