Automatisierte Logik und Programmierung

Einheit 4

Interaktive Beweisführung

1. Dienstleistungen des **Nuprl** Systems
2. Hinweise zum praktischen Arbeiten

Wozu interaktive Beweiser

- **Beweisführung ist sehr schematisch**
  - Vorgabe eines Beweisziels
  - Iteratives Anwenden von Beweisregeln und Erzeugen von Teilzielen
  - Kreativität liegt in Wahl geeigneter Regeln
- **Beweisführung von Hand ist ineffektiv**
  - Viel Schreibarbeit, mühsam, fehleranfällig
- **Regelanwendung ist programmierbar**
  - Vergleiche Beweisziel mit Hauptziel der Regel
  - Instantiiere Parameter in Regel und generiere Teilziele
- **Ein erster Schritt zur Beweisautomatisierung**
  - Benutzer konstruieren Beweise interaktiv durch Angabe der Regeln
  - Spart unnötige Schreibarbeit
  - Ermöglicht Experimente mit verschiedenen Beweisansätzen
  - Ermöglicht Programmierung von Beweisstrategien (als Taktiken)

Merkmale interaktiver Beweiser

- **Beweiseditor**
  - Benutzer formuliert Problem
  - Benutzer navigiert durch (unvollständigen) Beweisbaum
  - Benutzer gibt Beweisregeln ein und System generiert Unterziele
- **Bibliothek**
  - Verwaltung von Theoremen und zugehörigen Beweisen
  - Verwaltung von Definitionen und ggf. anderen Objekten
  - Mechanismen zum Erzeugen, Löschen, Anordnen von Objekten
  - Aufruf von Spezialeditoren zur Modifikation von Objekten

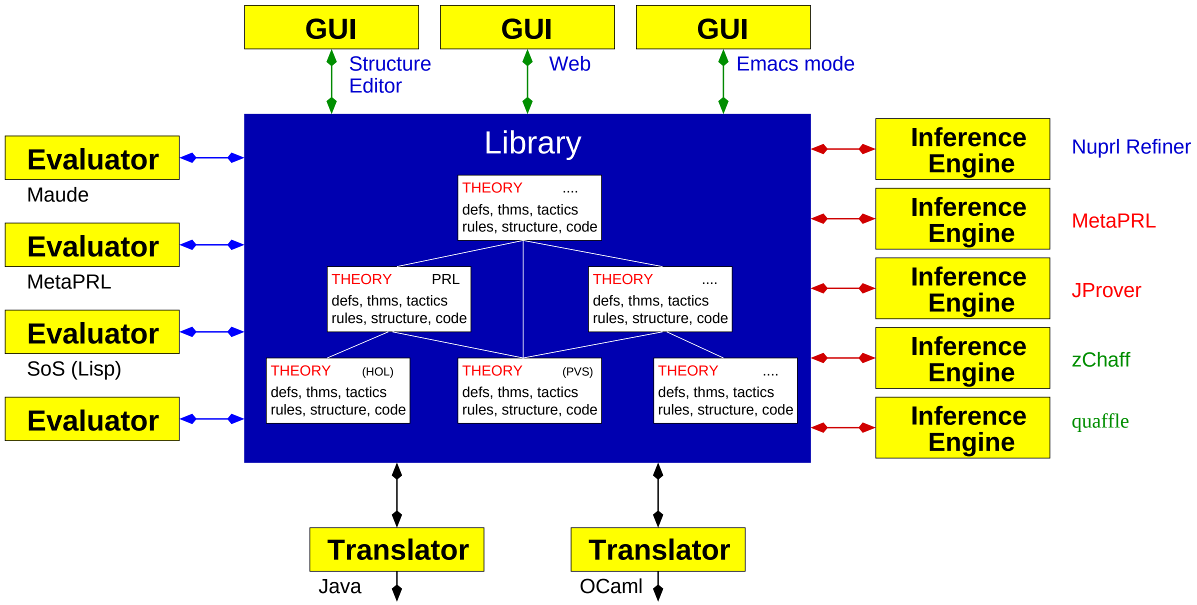
Viele weitere Merkmale möglich

Das interaktive Beweissystem Nuprl

"Proof Refinement Language, Version ν"

- **Anfänge liegen mehr als 20 Jahre zurück** (1984)
  - Nuprl 1 (Symbolics): Beweissystem für intuitionistische Typentheorie
  - Nuprl 2: Unix Version
- **Nuprl 3: Strategische Beweisführung** (1987)
  - Taktiken unterstützen Beweise ungelöster mathematischer Probleme
    - Girard's Paradox, Higman's Lemma
- **Nuprl 4: Logisches "Framework"** (1992)
  - Explizite Repräsentation des Kalküls im System
  - Systemkonzept unterstützt vielfältige Logiken und Notationen
  - Hohe Flexibilität ermöglicht Verifikation & Optimierung von Software
    - Logikschaltungen, SCI cache coherency protocol
    - Ensemble Kommunikationssystem (100000 Zeilen ML Code)

Nuprl 5: Kommunizierende Prozesse



- **Auftrennung der Systemkomponenten**
  - Bibliothek: Verwaltung des Wissens
  - Refiner: Ausführung der Inferenzmechanismen
  - Editor: Benutzerinterface
- **Plattform für Kooperation von Beweissystemen**
  - Einbindung externer Beweisprozeduren als weitere Systemprozesse
  - Anbindung neuartiger Komponenten möglich

Wichtige Dienstleistungen für Benutzer

- **Strukturеditor für Bearbeitung von Termen**
  – Mehr als ein Texteditor ... erkennt syntaktische Struktur von Termen
  – Neueingabe von Termnamen erzeugt Termstruktur im Display
  – Benutzer kann Notation und Layout von Termen modifizieren
- **Beweiseditor für Top-Down Beweise**
  – Strukturierte Bearbeitung von Beweisbäumen
  – Eingabe von Beweisregeln erzeugt die noch offenen Teilziele
  – Unterstützung für (benutzerdefinierbare) Beweistaktiken
  – Unterstützung für "Schmierzettelbeweise" und vieles mehr
- **Navigator für Navigieren durch Bibliothek**
  – Visuelle Navigation, Suche, Öffnen von Theoremen, Definitonen, ...
  – Erzeugen und modifizieren von Bibliotheksobjekten und Theorien

... und vieles andere mehr (mehr dazu im Sommersemester)
  – Komplexe Implementierung, nicht ganz leicht zu erlernen

Arbeiten mit Nuprl: Navigator

```
- TERM: Navigator
MkTHY*  OpenThy*  CloseThy*  ExportThy*  ChkThy*  ChkAllThys*  ChkOpenThy*
CheckMinTHY*  MinTHY*  EphTHY*  ExTHY*

Mill*  ObidCollector*  NameSearch*  PathStack*  RaiseTopLoops*
PrintObjTerm*  PrintObj*  MkThyDocObj*  ProofHelp*  ProofStats*  showRefEnvs*  FixRefEnvs*
CpObj*  reNameObj*  EditProperty*  SaveObj*  RmLink*  MkLink*  RmGroup*

ShowRefenv*  SetRefenvSibling*  SetRefenvUsing*  SetRefenv*  ProveRR*  SetInOBJ*
MkTHM*  MkML*  AddDef*  AddRecDef*  AddRecMod*  AddDefDisp*  AbReduce*  NavAtAp*
Act*  DeAct*  MkThyDir*  RmThyObj*  MvThyObj*

↑↑↑↑  ↑↑↑  ↑↑    ↑  ←  <>
↓↓↓↓  ↓↓↓  ↓↓    ↓  →  ><

Navigator: [num_thy_1; standard; theories]

Scroll position : 0

List Scroll : Total 159,  Point 0,  Visible : 7
 ---------------------------------------------
 -> CODE  TTF  RE_init_num_thy_1
    COM   TTF  num_thy_1_begin
    COM   TTF  num_thy_1_summary
    COM   TTF  num_thy_1_intro
    DISP  TTF  divides_df
    ABS   TTF  divides
    STM   TTF  divides_wf
 ---------------------------------------------
```

– Navigieren durch Bibliothek mit Maus und Pfeiltasten
– Öffnen von Objekten mit passendem Editor (Mausclick oder Pfeil)
– Ausführen von Bibliothekskommandos mit Buttons
– Viele weitere Dienstleistungen

Arbeiten mit Nuprl: Termeditor

Aktiv, wenn Cursor in einem Termslot ist

`not`	$\neg A$
`and`	$A \wedge B$
`or`	$A \vee B$
`implies`	$A \Rightarrow B$
`all`	$\forall x{:}T.\, A$
`exists`	$\exists x{:}T.\, A$

- **Eingabe neuer Terme**
  - Eingabe des Termnamens erzeugt Template mit Subtermslots
  - z.B. Eingabe von `all` ↵ erzeugt
    $\forall$|`[var]:[type]. [prop]`
  - Unbekannte Namen werden als Variablen interpretiert

- **Modifikation von Termen**
  - Navigation im Syntaxbaum mit Emacs-ähnlichen Kommandos
  - Variablennamen und Zahlen können wie Text verändert werden
  - Teilterme müssen gelöscht und neu erzeugt oder kopiert werden

Arbeiten mit Nuprl: Beweiseditor

- PRF: not_over_and

```
# top
∀A,B:ℙ. (((¬A) ∨ (¬B)) ⇒ (¬(A ∧ B)))

BY allI

# 1

1. A:ℙ
⊢ ∀B:ℙ. (((¬A) ∨ (¬B)) ⇒ (¬(A ∧ B)))

BY |
```

- **Eingabe und Speicherung von Beweiszielen**
  - Leeres Theorem enthält Termslot `[goal]` für Beweisziel
  - Beweisziel wird bei erstmaliger Eingabe in Bibliothek gesichert
  - Nachträgliche Änderungen müssen explizit gesichert werden
- **Beweisführung**
  - Benutzer gibt Beweisregeln oder Taktiken im Regelslot an
  - Refiner kann synchron oder asynchron aufgerufen werden
  - Ergebnis wird unmittelbar in Bibliothek gespeichert
  - Editor zeigt erzeugte Teilziele an und erzeugt neue Regelslots
  - Unterstützung für unvollständige Beweise und "Schmierzettelbeweise"

Arbeiten mit Nuprl: Beweistaktiken

- **Benutzerprogrammierbare Beweisstrategien**
  - Planung und Suche von Beweisen
  - Strukturierung von Beweisen (Verstecken überflüssiger Details)
  - Abgeleitete Inferenzregeln für benutzerdefinierte Theorien
  - Austesten komplexer Beweis-/Syntheseverfahren in sicherer Umgebung
- **Formalisiert als Metalevel-Programme**
  - Formale metasprachliche Ausdrücke (ML) steuern Anwendung von Regeln
  - Einfache Komposition von Regeln und Taktiken durch Tacticals

t_1 `THEN` t_2:	Wende t_2 auf alle von t_1 erzeugten Teilziele an
t `THENL [`$t_1;\ldots;t_n$`]`:	Wende t_i auf das i-te von t erzeugte Teilziel an
t_1 `ORELSE` t_2:	Wende t_1 an. Falls dies fehlschlägt, wende t_2 an
`Repeat` t:	Wiederhole Taktik t bis sie fehlschlägt
`Try` t:	Wende t an, Bei Fehlschlag lasse den Beweis unverändert
`Complete` t:	Wende t nur an, wenn der Beweis vollständig wird
`Progress` t:	Wende t nur an, wenn ein "Fortschritt" erzielt wird

- **Taktiken sind immer korrekt**
  - Taktikbeweis ist expandierbar in Beweis mit elementaren Regeln

Beispiel: Ein einfacher taktischer Beweiser

Sortiere Regelanwendungen nach Aufwand für Beweissuche

```
let simple_prover = Repeat
                    (       Hypothesis
                    ORELSE contradiction
                    ORELSE InstantiateAll
                    ORELSE InstantiateEx
                    ORELSE conjunctionE
                    ORELSE existentialE
                    ORELSE nondangerousI
                    ORELSE disjunctionE
                    ORELSE not_chain
                    ORELSE iff_chain
                    ORELSE imp_chain
                    );;
letrec prover = simple_prover
                THEN Try (        Complete (orI1 THEN prover)
                           ORELSE (Complete (orI2 THEN prover)))
                ;;
```

Arbeiten mit Nuprl – praktisch

- **Installieren auf eigenem Rechner**
  - Nuprl 5 System und Manual ist auf Webseiten verfügbar
    - Immer noch Experimentalsoftware mit einigen Quirks
    - Manual müsste in einigen wichtigen Punkten überarbeitet werden
  - Implementierung in CMU Common Lisp für Linux
    - Systemanforderungen: $\geq$512MB RAM, 1GB freier Plattenplatz
    - Mehr Prozessorleistung und RAM ist besser
- **Zugriff auf laufende Version über VNC**
  - Nuprl 5 läuft auf Servern der Theoriegruppe
  - Benutzer benötigen eigenen Editorprozess
  - Bibliothek und (mehrere) Refiner laufen im Hintergrund
  - Anleitungen zum Aufruf werden demnächst bereitgestellt