Margrit Dittmann

March 5, 2015

Die zur Auswahl stehenden Systeme sind NuPRL, Coq und Idris, wobei die ersten beiden auf ML basieren und Idris auf Haskell. In NuPRL und Coq gibt es bereits erste Realisierungen von Automaten.

1 NuPRL

Kreitz 1986 - Basiselemente: Types und Members of Types - Types sind in Universen angeordnet$U_1 \subseteq U_2 \subseteq U_3$ - atomare Typen: int, atom, void (Typ ohne Members) – A list ⇒ Liste von Elementen von A – A#B ⇒ kartesisches Produkt – A| B ⇒ disjunkte Vereinigung – A→B ⇒ Funktion von A nach B - Funktionale Objektsprache - Top-Down Verfahren - automata Bibliothek beinhaltet eine Reihe von Taktiken Kreitz 2000 - User Interface - interaktive und Taktiken (meta-level code) basierte Argumentation - automatische Tools: Entscheidungsprozeduren, Theorembeweiser, Beweisplaner, Modelchecker, Rewrite Enginges - erweiterbare Bibliothek -

> The Nuprl LPE (logical programming environment) is an open , distributed architecture that integrates all its key subsystems as independent components and, by using a flexible knowledge base as its central component, supports the interoperability of current proof technology.

- verschiedene Nutzer können parallel durch verschiedene Editoren am gleichen Problem arbeiten - Struktur: – Wissensbasis: ** Änderungen werden umgehend in die Bibliothek üertragen, dadurch gehen im Falls eines Systemausfalls keine Daten verloren ** bei Änderungen an Objekten werden die objekte nicht überschrieben sondern es werden neue Versionen davon erstellt ** folgt den gleichen Protokollen wie Datenbanken ** ältere Versionen von Objekten können wieder hergestellt werden und so kann man auch mehrere Beweis eines Theorems erstellen ** unterstützt dependency tracking ** Bibliothek schreibt keine vordefinierte Struktur (externe Repräsentation) vor. Die sichtbare Struktur wird durch Strukturobjekte generalisiert, die explizit in der Bibliothek angegeben sind. ** Aus dem gleichen Grund gibt es bei Namen eine Unterteilung in Namen, die der Nutzer vergibt und die das System vergibt. Daher können verschiedene Objekte auch ohne Probleme gleich benannt werden. – Benutzeroberfläche ** Kommunikation mit der Wissensbasis durch senden und empfangen von abstrakten Termen ** Funktionalität eines Strukturediteors ** ist imstande Objekte als Kommandos zu interpretieren ** Nutzer kann Definitionen von logischen Ausdrücken und Taktiken rekonstruieren oder den Editor anpassen mit Buttons für oft benutzte Befehle – Inferenz Maschine ** Beweise werden geprüft durch die Ausführung von ML Code ** gradlinig –¿ könnte mit

den ganzen verschiedenen Displayoptionen zu unübersichtlich werden –¿ Code nicht unbedingt intuitiv lesbar –¿ bietet viele Schnittstellen, jedoch für Einsteiger in meinen Augen zu komplex –¿ System läuft nur über eine virtuelle Maschine und es ist keine lauffähige Version ohne Probleme zu bekommen

2 Coq

- basiert auf abhängige Typentheorie - Läuft auf allen Betriebssystem - übersichtliche Coqide und viele Turotials auch zur Installation online verfügbar - aktive Community

3 Idris

Bei Idris gab es bei der Installation unter Ubuntu keine Probleme, jedoch unter MacOS 10.6.8. Hierbei müssen Pakete teilweise einzelen installiert werden, wobei allerdings die benötigten Abhängigkeiten nicht mehr gewährleistet werden können. - ausführbarer Code wird generiert - funktionales Interface wechles die Interaktion mit einer externen C Bibliothek ermöglicht - auf Haskell Basis - Datentypen ähnlich wie in Haskell - Funktionen werden durch Pattern Matching implementiert ** Name : Type -¿ Type -¿ Type ** Name Pattern ** Name Var=Funktionsbeschreibung – Funktionsnamen können sowohl mit Groß-, als auch mit Kleinbuchstaben beginnen - Umgang mit Zahlen als Eingabe genauso als Typ von Nat - Vektoren als abhängige Typen (vom Typ Liste mit Länge) - verständliche Fehlermeldungen - endliche Mengen - Umgang mit Dateien als Eingabe mglich (lesen/schreiben...) - Einrückung muss beachtet werden - anonyme Funktionen (\x =¿ val) - Maybe um optionale Werte zu beschreiben - Sigma Types als anhängige Paare (2. Element vom Paar hängt vom 1. Element ab) - _ als Platzhalter - Type Classes um typüebrgreifende Funktionen zu definieren - Namen können in verschiedenen Modulen mehrfach verwendet werden, da diese modulspezifisch sind - import referenziert Dateinamen und durch einen Punkt können Pfade angegeben werden - Vererbung: Module können andere Module verwenden, je nach Definition (abstract, public, private) ** public: Name und Definition werden exportiert ** abstract: Name wird exportiert ** private: weder Name noch Definition werden exportiert - Parameter als separate Blöcke, die auch abhängige Typen mit impliziten Argumenten sein können - Mischung aus Coq und Agda da sowohl interaktives Beweisen als auch Taktiken verwendet werden können

4 Coq und NuPRL

- Beide Systeme bieten Einführungs-, Gleichheits- und Eliminationsregeln an . - interaktive Beweissysteme auf Basis vom ML - Hypothesenliste zur Darstellung der noch ausstehenden Beweise, um das Ziel zu beweisen - Verwendung von Taktiken, um Beweise zu führen - Verwendung von Taktiken um eine bessere Lesbarkeit zu ermöglichen

5 Entscheidung

Ein wichtiger aspekt für die spätere Anwendung ist eine einfache Installation des zugrundeliegenden Systems unabhägig vom Betriebssystem.