PL0 Compiler

Abschlusspräsentation

Benjamin Dittes & Matthias Lange

Rückblick (1)

- Compiler für eine allgemeine Grammatik (Spezifikation durch XML)
  - tabellengesteuertes Verfahren
  - Zielcode durch Semantikbefehle in der Grammatik definieren
  - Interpreter für abstrakte Maschine (z.B. AM_0)

Rückblick (2)

- Definition der Toolbox-Funktionen
  - vollständige Grammatikdefinition inkl. Semantikbefehlen
  - Codeerzeugung für abstrakte Maschine
  - kein Interpreter oder Ähnliches

Ziel heute

- Vorstellung der Toolbox-Funktionen
- Vorstellung unseres “Interpreter”-Konzepts
- Demonstration des Gesamtprojekts

Toolbox-Funktionen

EnterLevel(name) LeaveLevel()	legt Semantiklevel “name” an und speichert die Zeilennummer verlässt das aktuelle Level und betritt dessen Vater
Define(name, type) CheckDefine(name, type1:type2: ...)	legt Variable “name” des Typs “type” an prüft die Sichtbarkeit der Variable “name” vom Typ “type1”, “type2”, “...”
SetJumpMark(name) ResolveMarkToLine(name) ResolveLevelToLine(name)	verknüpft “name” mit aktueller Zeilennummer liest registrierte Zeilennummer für “name” aus liest registrierte Zeilennummer aus der
MakeAddress(name) ResolveAddress(name)	erstellt neue Adresse (beginnend bei 0) für “name” liest registrierte Adresse für “name” aus

Toolbox-Funktionen

ResolveLevelToString(name, sep) ResolveCurrentLevelToString(sep)	erstellt qualifizierten Namen mit dem Separator “sep” in der Semantik-Node “name” wie oben, nur in der aktuellen Semantik-Node
MakeAddressForCurrentLevel(Sep) ResolveLevelToAddress(Name,Sep) ResolveCurrentLevelToAddress(Sep)	erstellt Adresse für qualifizierten Namen “sep” liest Adresse für qualifizierten Namen “sep” aus Semantik-Node “name”
All(Commands,Type1:Type2:...) AllR(Commands,Type1:Type2:...)	Dupliziert 'Commands' für alle Variablen-einträge von eines der Typen genau wie All(...), nur umgekehrte Reihenfolge
@name _ $ {i}	wird durch eindeutigen Namen ersetzt Leerzeichen Newline i-tes Symbol der Syntaxoption

Interpreter

- Nutzung des Interpreters des PL0C-Teams ???
- keine Festlegung auf bestimmten Zielcode, daher auch Transformationen möglich (z.B. PL/0 -> C++)
- Nutzung weiterer Compiler um lauffähiges Programm zu erzeugen (Plattformunabhängigkeit erreicht)

Demonstration