Seminar „Visuelle Sprachen“

27.05.2004

Joana Hois

Übersicht

- Herkunft und Ziele der Sprache
- Abstrakte Sprachkonzepte
- Konkrete Sprachkonzepte
- Datentypen
- Zeit-orientiertes Konzept
- Testen

Übersicht

- Herkunft und Ziele der Sprache
- Abstrakte Sprachkonzepte
- Konkrete Sprachkonzepte
- Datentypen
- Zeit-orientiertes Konzept
- Testen

Entstehung

- Research Language
- Vorgänger:

  Forms (1986-88), Ambler

  Forms/2 (89-90), Ambler & Burnett

- Forms/3 (1991), Margeret M. Burnett
  - Lucid Common Lisp
  - Später: Userinterface-Erweiterung

Zielgruppe

- End-User-Programmierer
  - Eigentliche Anwender
  - Programmier-Neulinge
- Professionelle Programmierer

- „gentle slope language“ [1]

- Kein spezieller Anwendungsbereich der Applikationen

Ursprung der Sprache

- Basiert auf Tabellenkalkulation („spreadsheet“)
- Folgt dem spreadsheet paradigm:
  - Berechnungen sind definiert durch Zellen (Tabelleneinträge) und ihre Formeln
  - Alan Kay‘s value rule

Vorherige Spreadsheet-Sprachen

- Starke Einschränkungen durch:
  - Geringe Datentypunterstützung (Bool, int, string)
  - Keine Abstraktionsmöglichkeiten (für Datentypen und Prozeduren)

- Forms/3:
  - soll nicht diesen Einschränkungen unterliegen
  - soll dabei aber das Spreadsheet-Paradigma nicht verletzen

Übersicht

- Herkunft und Ziele der Sprache
- Abstrakte Sprachkonzepte
- Konkrete Sprachkonzepte
- Datentypen
- Zeit-orientiertes Konzept
- Testen

Sprachkonzepte: Grundlagen

- Spreadsheet-Sprachen ähneln funktionaler Programmierung
  - Übergabe von Argumenten an Funktionen und Operatoren
  - Deklarativ

Sprachkonzepte: Grundlagen

- Spreadsheet-Sprachen sind aber auch unterschiedlich zu funktionaler Programmierung
  - In Forms/3: nur first-order functions
  - Continuous evaluation

Designziele von Forms/3

- Kombination von klassischem Programmieren und Tabellenkalkulationen
- Directness
- Direktes semantisches Feedback
  - automatisch
  - mit weiteren Auslösern

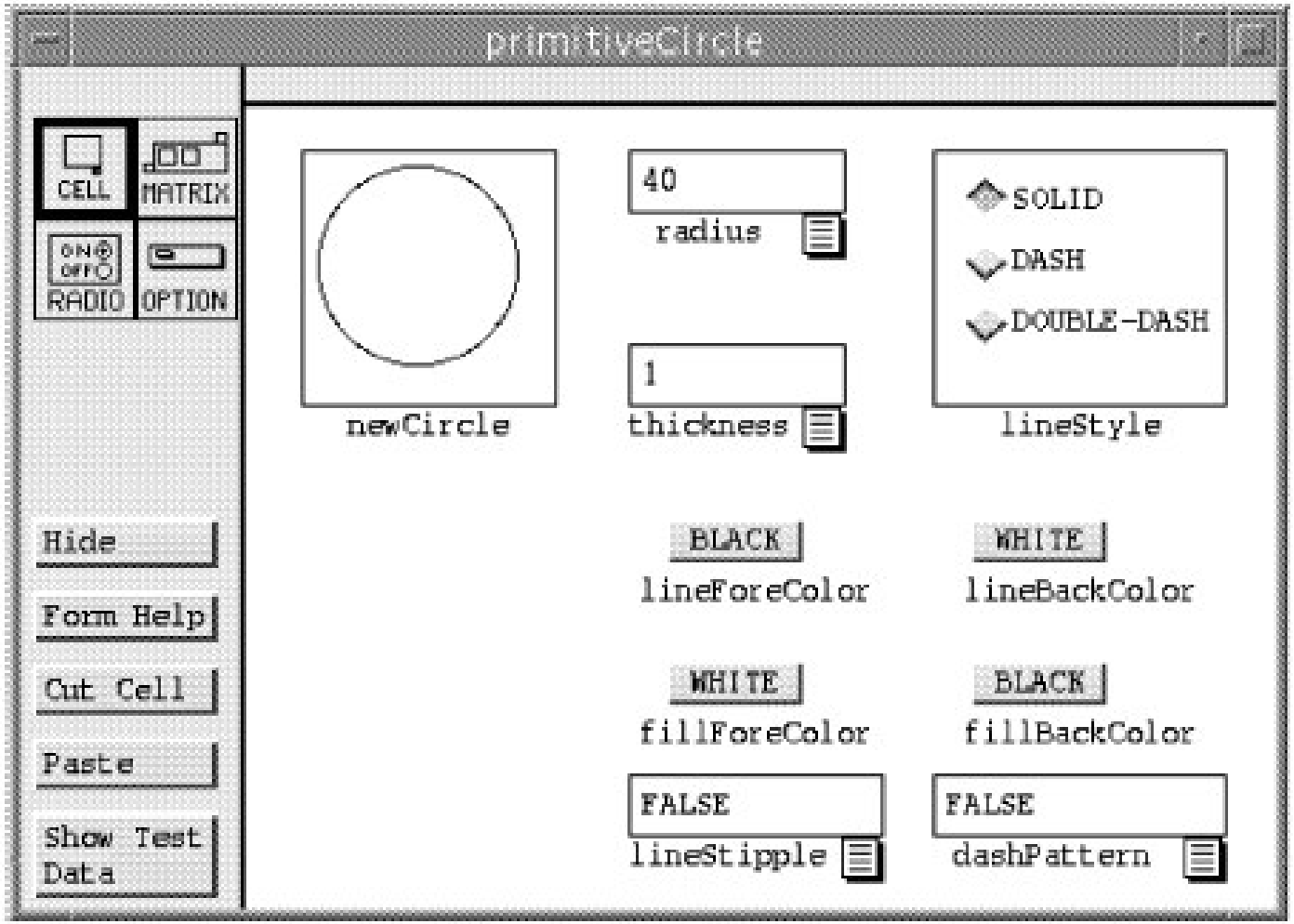
Übersicht

- Herkunft und Ziele der Sprache
- Abstrakte Sprachkonzepte
- Konkrete Sprachkonzepte
- Datentypen
- Zeit-orientiertes Konzept
- Testen

Sprachkonzepte: Basic Features

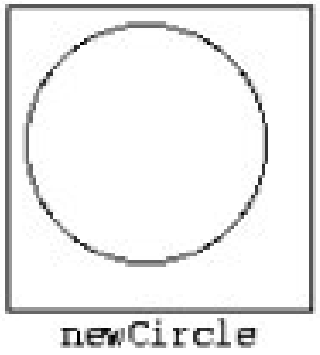
- Programm besteht aus forms (Formulare/Tabellen)
- Forms: Grundbaustein der Sprache
  - Subprogramm/Modul
  - Typdefinition
  - Besteht aus Zellen
  - Einzelne Zellen eines forms können flexibel angeordnet werden (dynamisches Raster)

Beispiel: Kreis

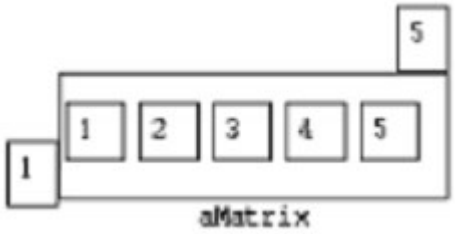


Zellen – 3 Varianten

- Einfache Zelle



- Dynamische Matrix



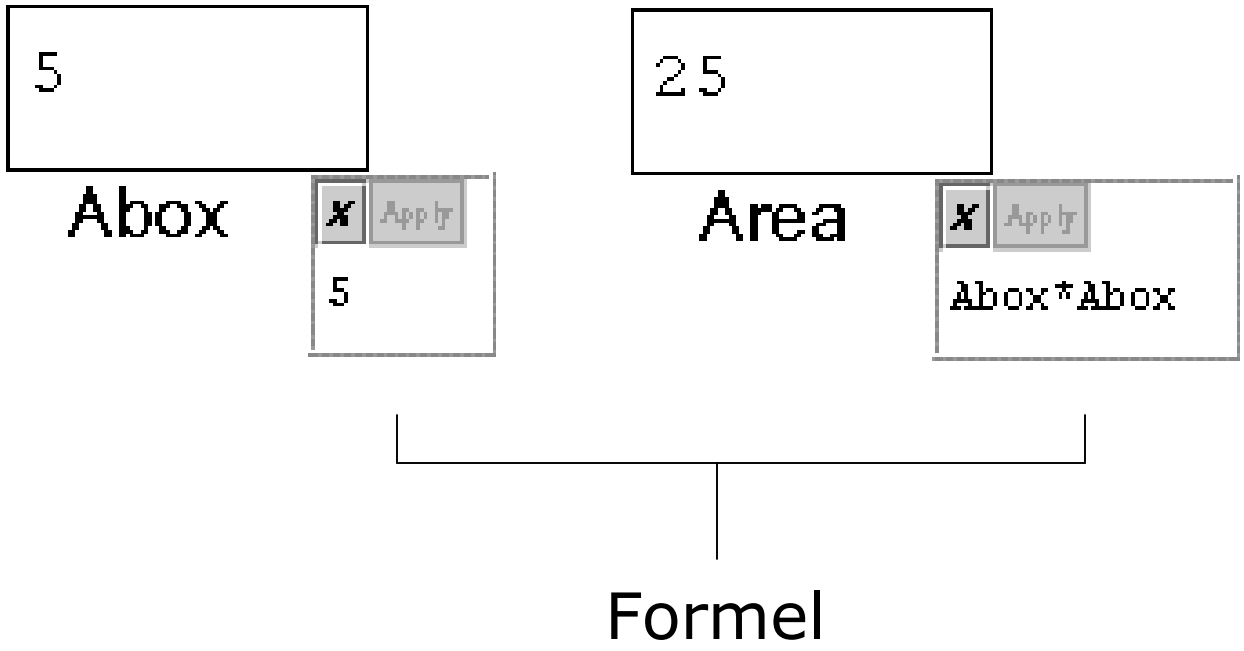
- Abstraktions-Box

Zell-Formeln

- Berechnen den Wert/die visuellen Attribute der Zelle
- Textuell
- Dynamische Typbindung

- Beliebig komplexe Ausdrücke
  - IF-THEN[-ELSE]
  - Rekursion
  - Beliebige Operatoren, auch selbst definierte
    - Unär und binär (Auswertung: links nach rechts)
  - Beliebige Schachtelung

Beispiel: Zellformel

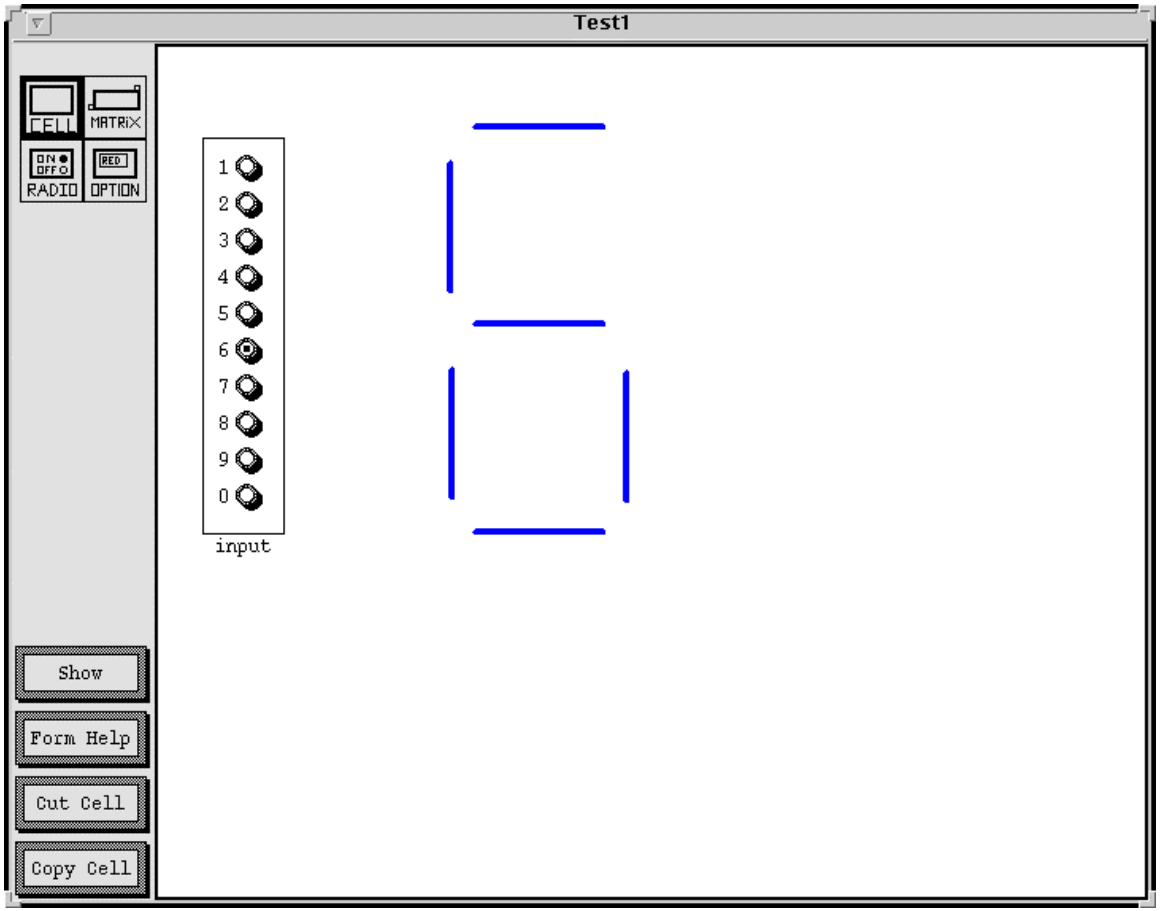


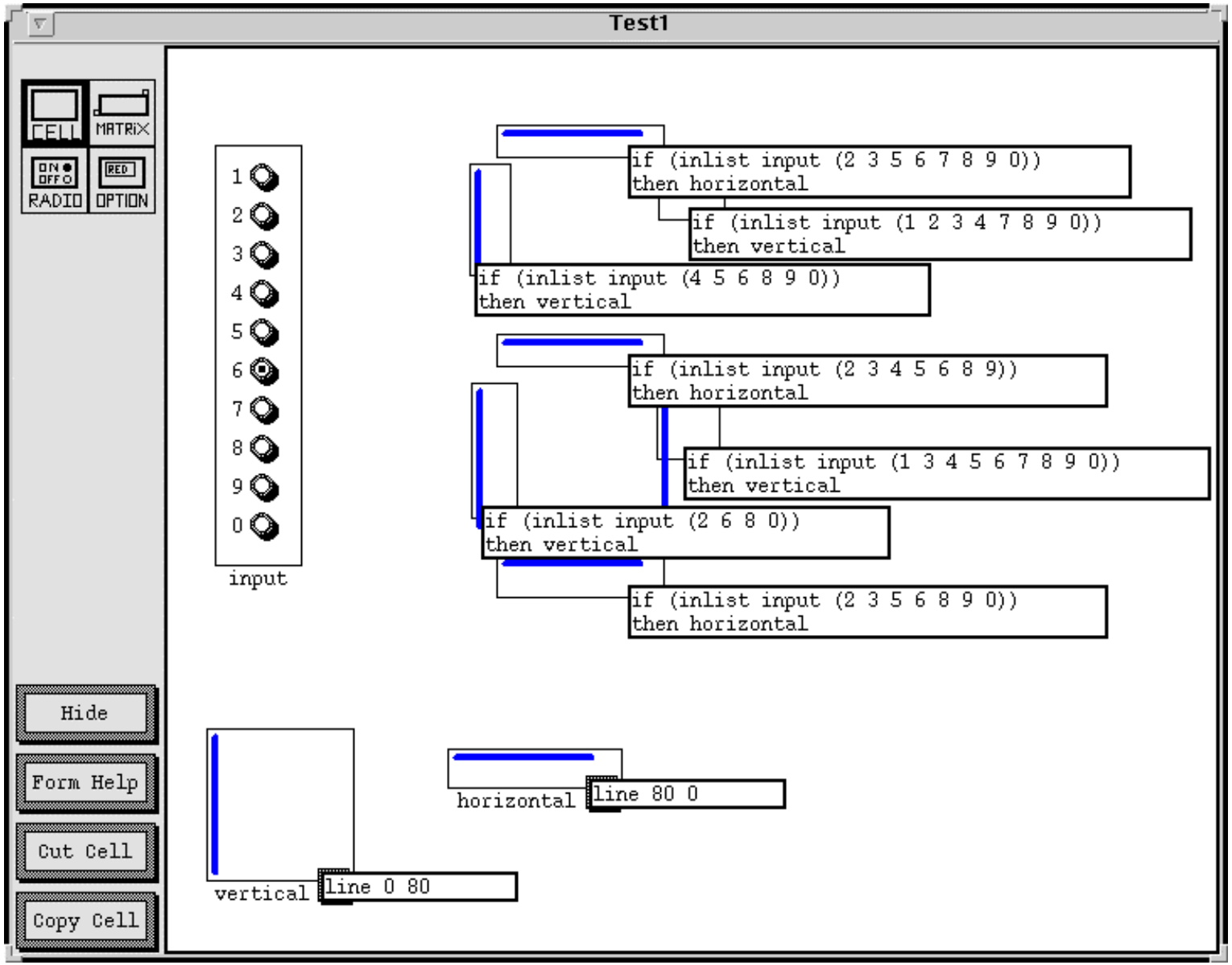
Basic Features - Specials

- Blank: Kein vergebener Wert

- Dynamisches Raster
  - wird dynamisch und zeitnah generiert
  - Pseudo-Referenzen (z.B. in Matrizen: I, J)
  - Referenzierung über regions möglich

Beispielprogramm: Single Digit LED




Test1
CELL
MATRIX
RADIO
OPTION
1
2
3
4
5
6
7
8
9
0
input
if (inlist input (2 3 5 6 7 8 9 0))
then horizontal
if (inlist input (1 2 3 4 7 8 9 0))
then vertical
if (inlist input (4 5 6 8 9 0))
then vertical
if (inlist input (2 3 4 5 6 8 9))
then horizontal
if (inlist input (1 3 4 5 6 7 8 9 0))
then vertical
if (inlist input (2 6 8 0))
then vertical
if (inlist input (2 3 5 6 8 9 0))
then horizontal
Hide
Form Help
Cut Cell
Copy Cell
horizontal
line 80 0
vertical
line 0 80

Übersicht

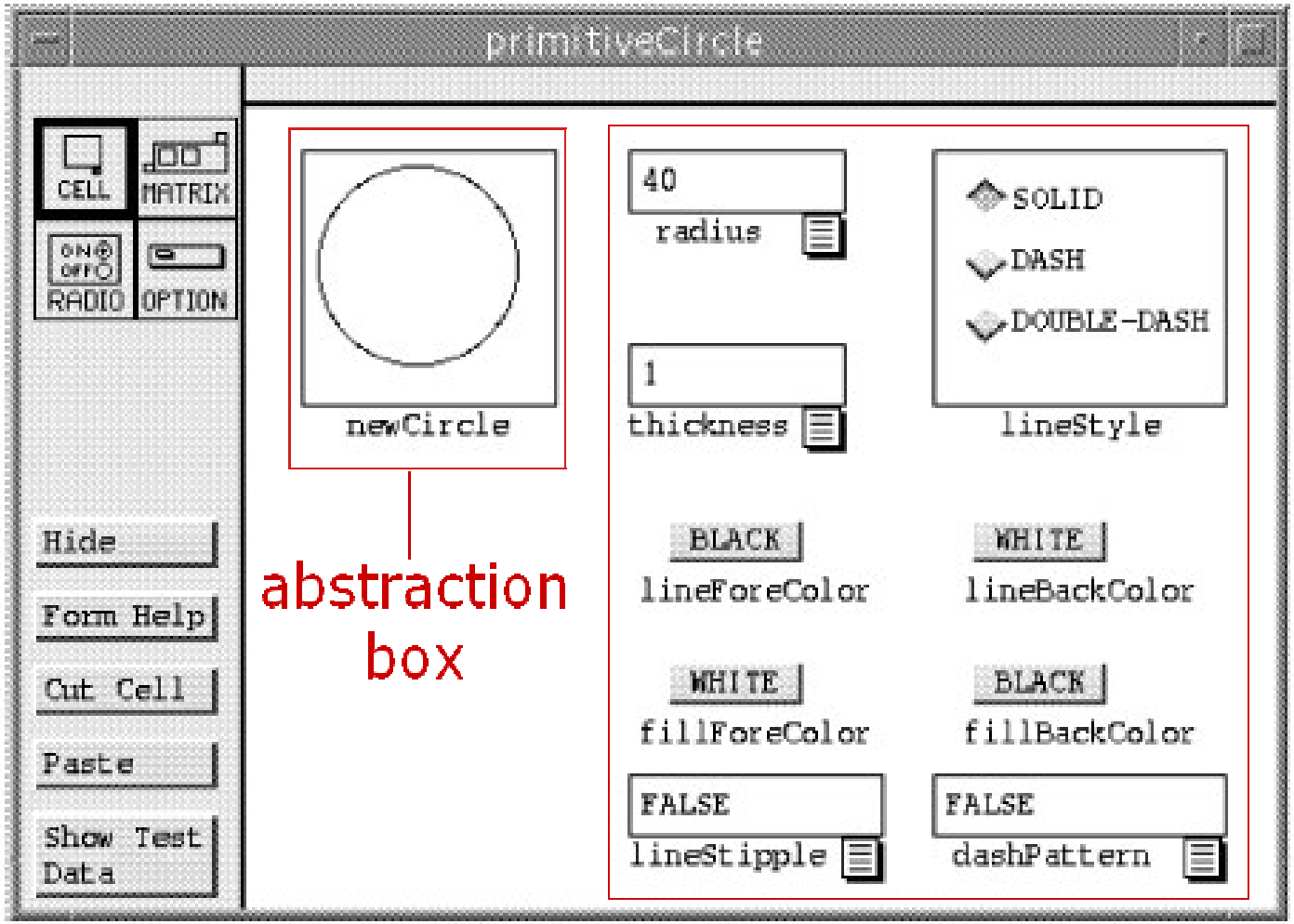
- Herkunft und Ziele der Sprache
- Abstrakte Sprachkonzepte
- Konkrete Sprachkonzepte
- Datentypen
- Zeit-orientiertes Konzept
- Testen

Datentypen

- Grunddatentypen: Bool, int, string
  - Keine Deklaration nötig

- Komplexe und user-definierte Datentypen:
  - Spezielle forms bestehend aus
    - abstraction box (hidden Komponenten)
    - image cell (graphische Repräsentation)
    - Weitere Zellen (Operationen, interaktives Verhalten)

Beispiel: build-in type



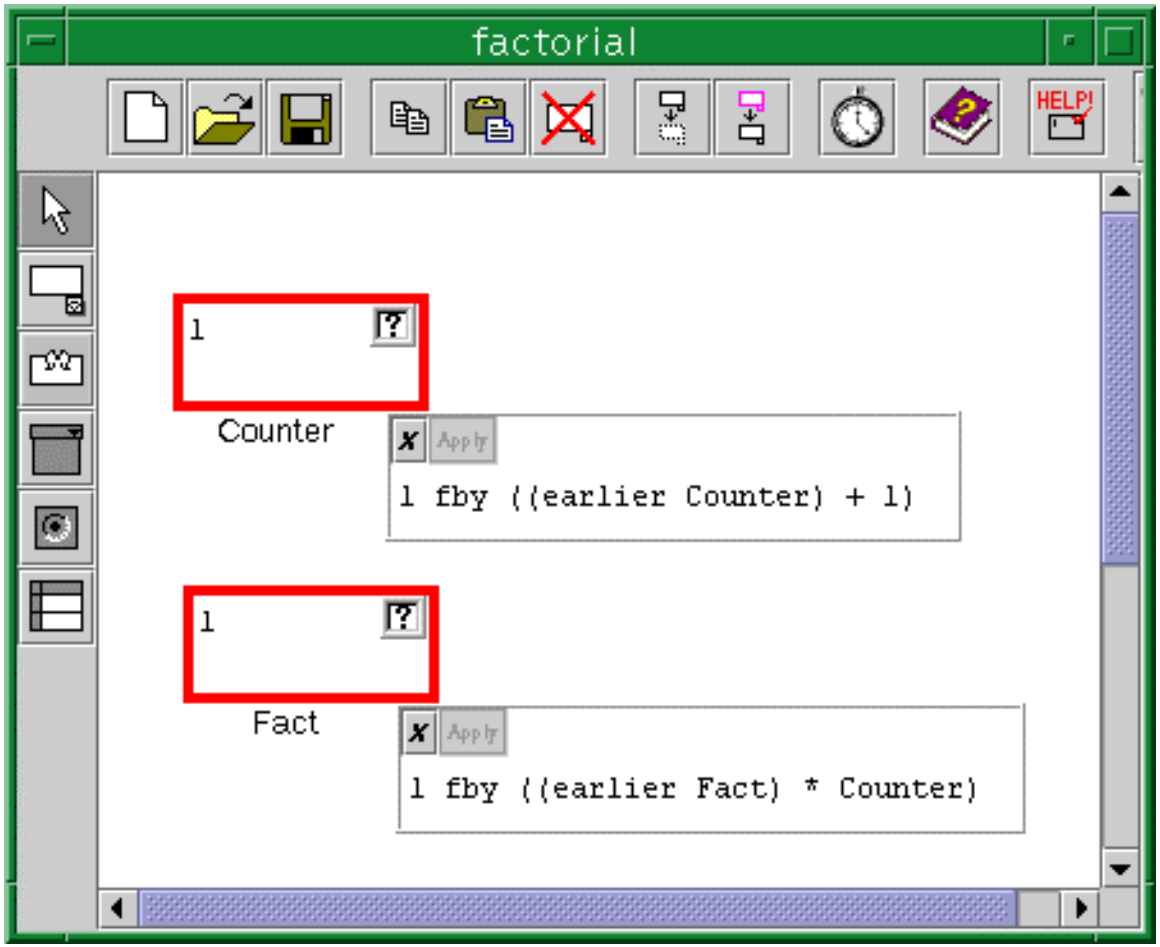
Übersicht

- Herkunft und Ziele der Sprache
- Abstrakte Sprachkonzepte
- Konkrete Sprachkonzepte
- Datentypen
- Zeit-orientiertes Konzept
- Testen

Zeit-orientiertes Konzept

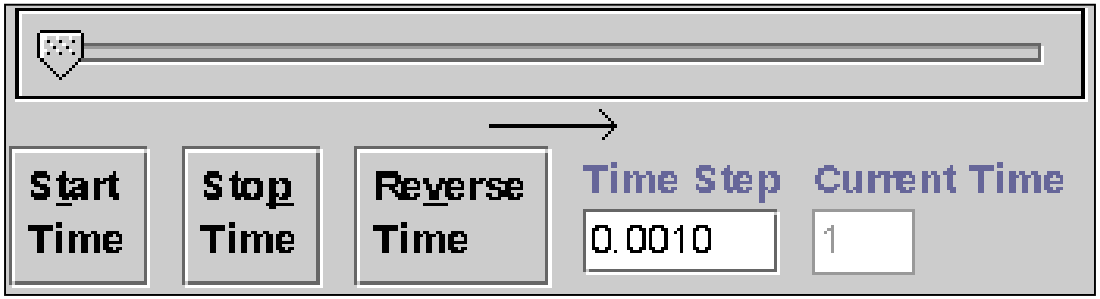
- Zellwerte sind über Zeiträume definiert
  - Dynamische Veränderung von Rasterinhalten (Werte und graphische Darstellung)
- Hängt nicht von „natürlicher Zeit“ ab
  - System-Clock, Events etc.
- Zwei Operatoren für Formulare:
  - earlier
  - fby (followed-by)

Beispiel: Fakultät

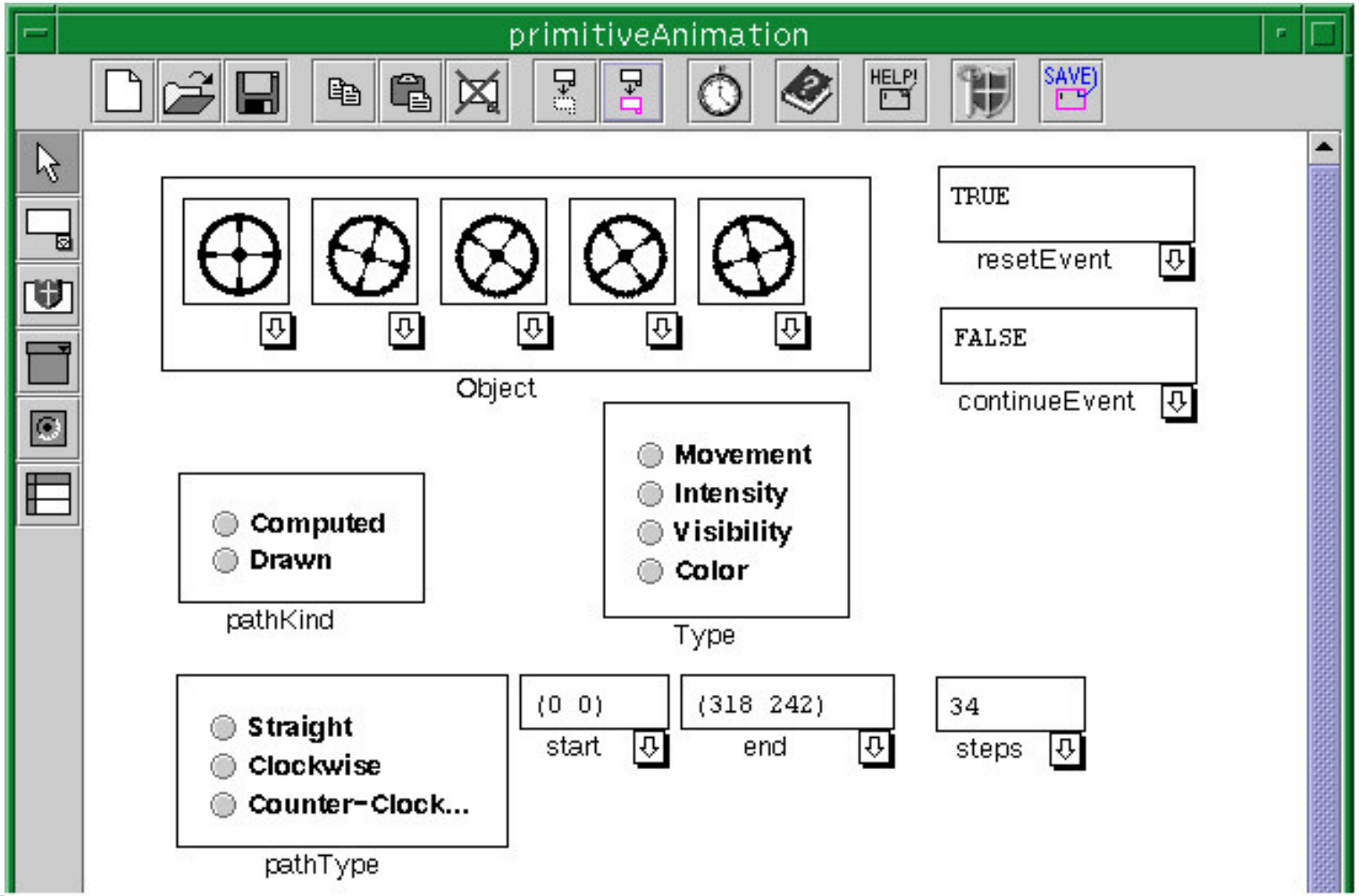


Time Travel

- Zeitliche Abfolge des Programms sichtbar
  - Animationen möglich
- Lokales Debugging
  - zu jedem Zeitpunkt können Zelleninhalte geändert werden

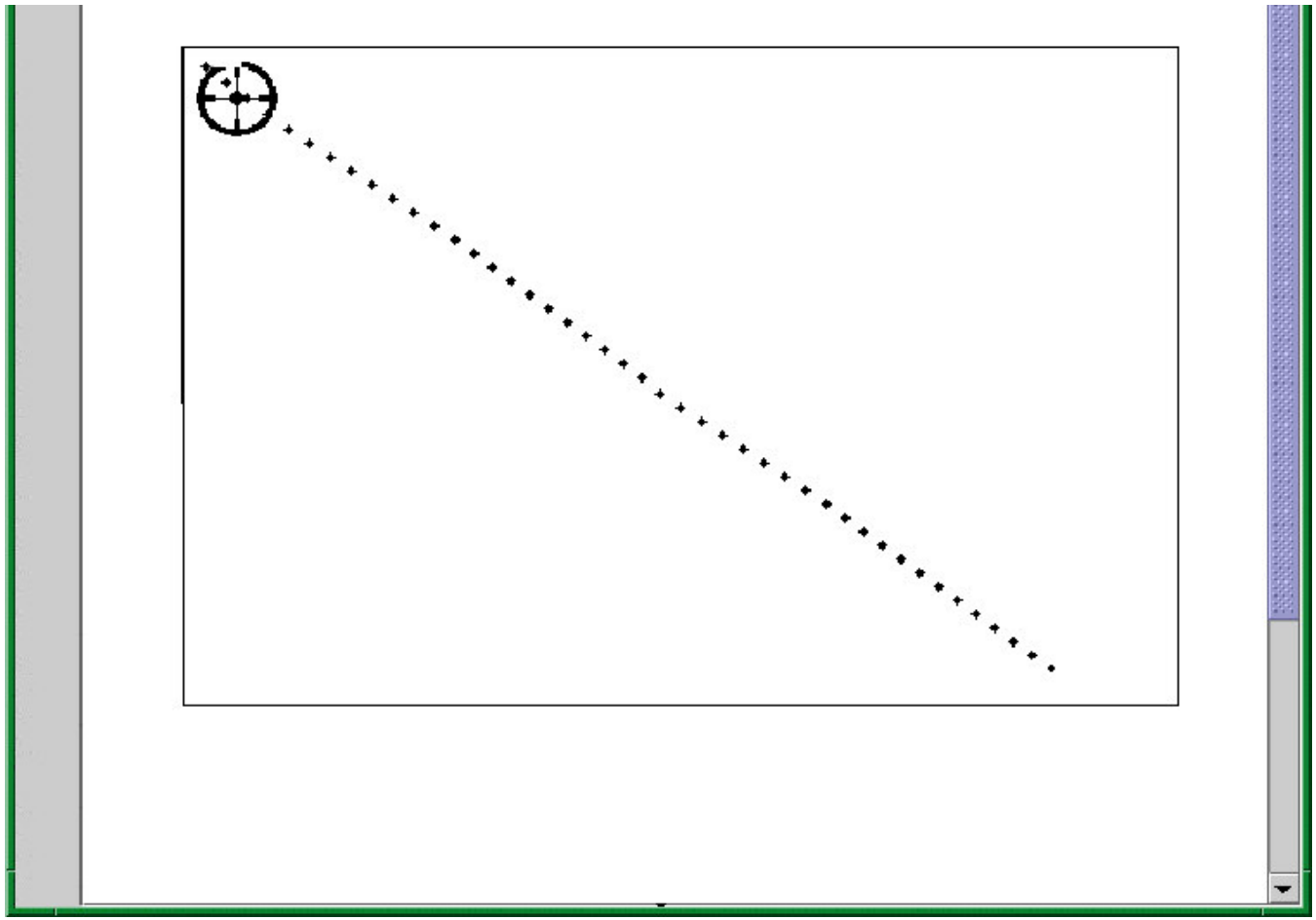


Beispiel: animiertes Rad



Forms/3

Beispiel: animiertes Rad



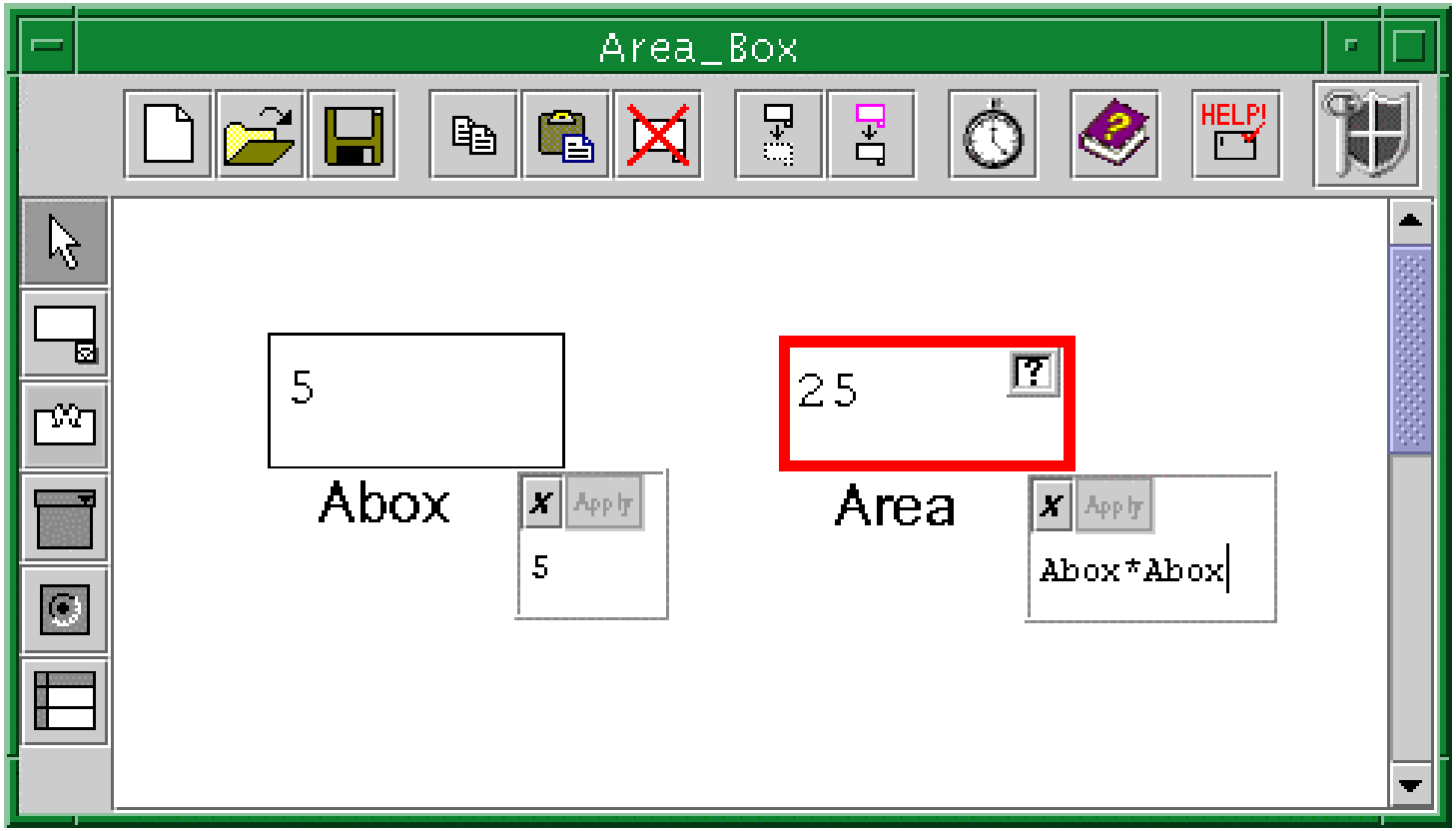
Übersicht

- Herkunft und Ziele der Sprache
- Abstrakte Sprachkonzepte
- Konkrete Sprachkonzepte
- Datentypen
- Zeit-orientiertes Konzept
- Testen

Testen

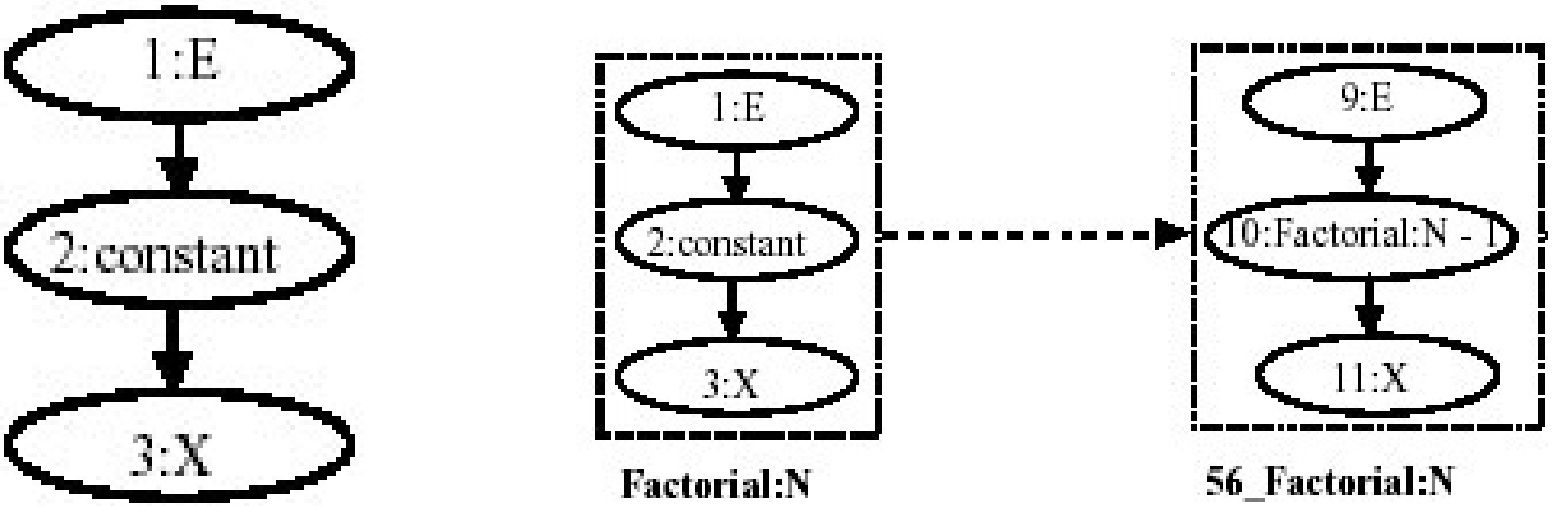
- Automatisiertes Testen wird von Forms/3 unterstützt
- Dynamisch
- WYSIWYT

Testen: graphische Darstellung



CRG – cell relation graph

- Zwei Komponenten:
  - formula graph
  - cell dependence edges



Resümee

- „Überladene“ Sprache
- Testsprache
- Kombination formaler und visueller Elemente
- Anwendungsbereich
- Testunterstützung

Literaturverzeichnis

[1] **Forms/3: A First-Order Visual Language to Explore the Boundaries of the Spreadsheet Paradigm (2001)**, M. Burnett et al., *Journal of Functional Programming* 11(2) S. 155-206

[2] **Forms/3-Homepage**:

http://web.engr.oregonstate.edu/~burnett/Forms3/forms3.html

[3] **What You See Is What You Test: A Methodology for Testing Form-based Visual Programs (1998),** *G. Rothermel et al., Proceedings of the 20th Int'l Conference on Software Engineering*, S. 198-207

[4] **Visually Testing Recursive Programs in Spreadsheet Languages (2001)**, M. Burnett et al., *IEEE Symposia on Human-Centric Computing Languages and Environments*