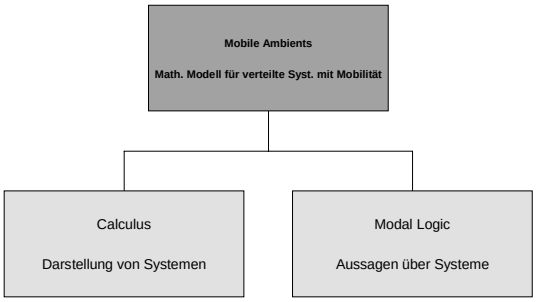
Einführung in Mobile Ambients

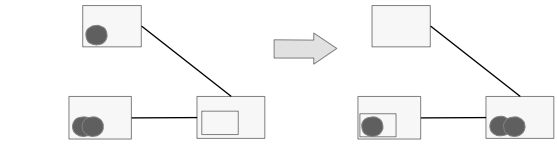
Seminar: Ubiquitous Computing WS 01/02

Was umfasst Mobile Ambients?



Was kann ich mit dem Calculus darstellen?

- Räumliche Konfigurationen
  - Mobile Computing (Laptops etc.)
  - Mobile Computation (Agents etc.)
- Zustandsveränderungen (zeitlich)



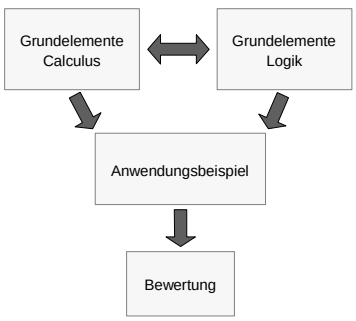
Mobile Agent Ambient

Was kann ich mit der Logik nachweisen?

- Beweise, dass bestimmte Zustände nicht erreicht werden
  - z.B. für Security
- Andere Zusicherungen

$$\text{Beh.}: \Box A \vdash \neg \Diamond B$$

Übersicht



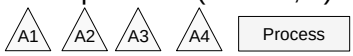
Grundlagen Calculus

- Prozesse und Ambients
- Umformungen
  - Congruence
  - Reduction Relations
- Kommunikation (Input/Output)

Komplexere Objekte können aus den obigen zusammengesetzt werden

Grundelemente des Caculus

Capabilities (actions,...)



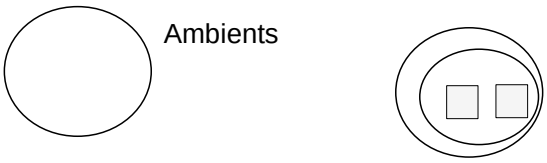


Abb. 1: Elements of Distributed Computing

Prozesse

- Abfolge von Aktionen (Capabilities)
  - abstrakt
  - konkret
- Sind i.d.R. in Ambients enthalten
- Können parallel zueinander laufen

Abstrakt: P

Konkret: $in\ n.P$

Parallele Prozesse: $P|Q|R$

Typische Capabilities

- *in n*: *subjective move*
- *out n*: Umkehrung zu *in*
- *open n*: löst ein Ambient auf, Inhalt bleibt aber bestehen

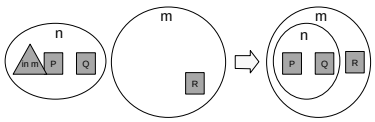


Abb. 2: Using the *in*-Capability

Ambients

- "[A] bounded place where computation happens"
- Kann sein:
  - Computer
  - Java Sandbox
  - Speicherbereich
  - .....

$$n[P]$$

Namen von Ambients

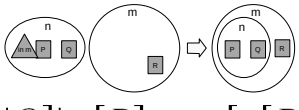
- Prozesse benötigen Namen der Ambients um mit ihnen zu interagieren
  - *open n*
- Namen können auf gewisse Bereiche beschränkt sein

$$(\upsilon n)n[Q]|P$$

P kann nicht *open n* enthalten, da n beschränkt!

Umformungsregeln

- Congruence: Prozesse sind bis auf triviale Syntax-Umformungen äquivalent
  - $P|Q \equiv Q|P$
- Reduction Relation: Zustandsübergang aufgrund der Capabilities



$$n[in\ m.P|Q]|m[R] \rightarrow m[n[P|Q]|R]$$

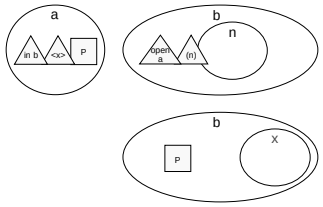
Kommunikation

- Es gibt Operationen
  - zum Einlesen von Information in Variablen, d.h. alle Stellen an der die Variable vorkommt, werden durch den Wert ersetzt
  - zum Ausgeben von Information
- Wird oft für Ambient-Namen verwendet
- Prozesse, die auf Input warten, blocken

$$\text{Output: } \langle x \rangle.P$$
$$\text{Input: } (n).n[0]$$

Kommunikation (2)

- Wir haben einen Äther, in dem Information so lange liegt, bis sie 'aufgesaugt' wird



Example of Inter-Ambient Communication

Modale Logik für Mobile Ambients

- Sehr komplex und umfangreich
- Hier sollen lediglich ein paar Grundkonzepte angesprochen werden
- **Wichtig:** unterstützt keine Namensrestriktionen!

Grundidee

- Im Prinzip eine modale Logik mit Prädikaten, und, oder, nicht, etc.
- Verschiedene Relationen, z.B.:
  - Satisfaction
  - Sequents
  - Inference
- Model Checker

Satisfaction

- Drückt aus, dass für einen Prozess gewisse Regeln gelten

$$P = a_k[0] \Rightarrow P_{k\triangleright j} \models a_k[\neg(j[T]|T)]$$
$$P = a_k[\langle x \rangle.0 | (a).in\ a.0] \Rightarrow P_{k\triangleright j} \models a_k[\neg(j[T]|T)]$$

Sequents

- Erlaubt 'Umformungen' von Eigenschaften
- "Wenn für einen Prozeß x gilt, dann gilt für ihn auch y"

$$a_k[\neg(j[T]|T)] \overset{(n[\neg 4])}{\vdash} \neg a_k[j[T]|T]$$

$$P = a_k[0] \Rightarrow P_{k\triangleright j} \models a_k[\neg(j[T]|T)]$$
$$P \models \neg a_k[j[T]|T]$$

Inference

- Wenn die Sequents auf der Linken alle gelten, dann gilt auch das Sequent auf der Rechten

$$(|\vdash) : A \vdash B; C \vdash D \succ A|C \vdash B|D$$

$$(Id) : \succ A \vdash A$$

$$(\Diamond T) : \succ A \vdash \Diamond A$$

$$A|B \overset{(|\vdash)(Id)(\Diamond T)}{\vdash} A|\Diamond B$$

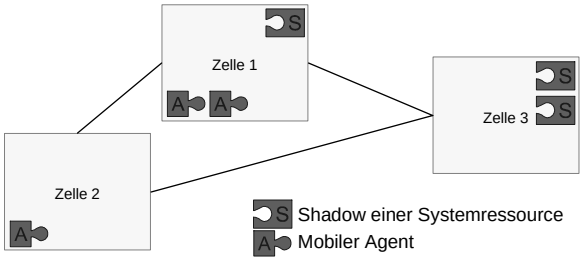
Model Checker Algorithmus

- Es gibt einen Satz an Regeln, mit denen Assertions algorithmisch bewiesen werden können
- z.B.:*Ist Prozeß P in zwei Ambients aufteilbar?*
- Nachteil: Funktioniert nur für ein Subset der Logik

$$P \models A \overset{\text{Model Checker Algorithmus}}{\rightarrow} \mathrm{T} \vee \mathrm{F} \qquad a[0] \models a[j[T]|T] \dashrightarrow \mathrm{F}$$

Beispiel: Anwendung auf HIVE

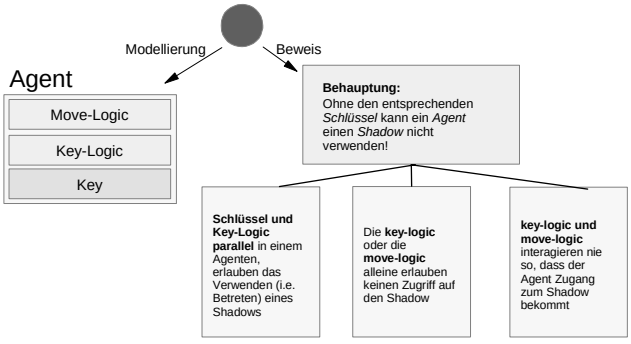
- Wurde bereits vorgestellt
- Läßt sich gut mit Mobile Ambients abbilden



Annahmen und Vereinfachungen

- Wir befinden uns in der Entwurfsphase
- Nur 1 Schritt für jeden Agenten
- Nur 1 Schlüssel pro Agenten
- Nur 1 Shadow pro Zelle
- Man kann von jeder Zelle in jede andere wechseln
- *Warum?* Beweis überschaubar halten

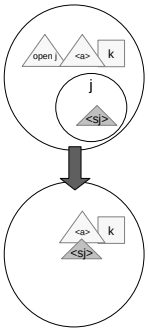
Beweisskizze



Beispielhafter Aspekt der Architektur: key-logic

$$\text{key-logic} \triangleq open\ j.(a).in\ a.out\ a.0$$

$$access\ j \triangleq\ !j[\underline{< s_j >}.0]$$



Beispielhafte Teilbehauptung bezgl. der key-logic

Wenn sich die key-logic in a_k befindet, und a_k parallel zu s_j positioniert ist, wird a_k nie in s_j eindringen.
Anders: Ohne key, ist die key-logic inert

$$\text{key-logic} \models (s_j \triangleright \neg\Diamond s_j[a_k[T]])@a_k$$

Beweisskizze:

$$\nexists P' : P \rightarrow^* P' \wedge P' \models s_j[a_k[T]]$$
$$P = a_k[open\, j.(a).in\, a.out\, a.0] | s_j[0]$$

Alle Permutationen ausprobieren

Pro und Contra

- **Logische Grundlage für Designentscheidungen**
- Hilft Probleme im Design zu erkennen, bzw. weist auf Gerfahrenherde hin
- Basiert auf bestehenden Konzepten
- **Komplexere Beweise können auf anderen basieren**
- **Es gibt einen Modelchecker-Algorithmus**
- **Modelliere ich die Wirklichkeit oder meinen Wunsch?**
- Extrem kompliziert
- Teilaspekte sind noch nicht erforscht
- **Keine Aussagen über Vergangenheit möglich (Sometime)**
- Bruch zw. Logik und Calculus
- Keine **Name Restrictions**