Temporales Schlussfolgern

Information aus Vergangenheit (Antezendent) → Konsequenz für Gegenwart und Zukunft

Ablauf:

1. update history durch empfangene messages
2. testen welche Regeln jetzt erfüllt sind
3. Ausführen der Regeln → erzeugt Bedingungen → diese müssen für Zustandsänderung erfüllt sein → Priorisierung der Handlungsmöglichkeiten → nicht erfüllte Änderungen werden im nächsten Durchlauf bearbeitet

Beispiele:

- Gestern war Mittwoch:
  - ● currentDay (Mittwoch)
- Irgend wann lernen wir:
  - ◇ study (we)
- Es gibt Freibier bis das Bier alle ist:
  - beer (free) U beer (empty)

○ φ : φ ist „morgen" wahr
■ φ : φ gilt immer in der Vergangenheit
φ W Ψ : φ gilt solange nicht Ψ

- Früher war alles beser: ■ allesbesser
- Morgen ist Wochenende: ○ wochenende

Concurrent MetateM

Grundidee: System aus untereinander kommunizierenden Agenten.

Jeder Agent erzeugt sein Verhalten mittels temporalen Schließens.

Aufbau eines Agenten:

└ Andere Gruppe?

agent id (= Name)

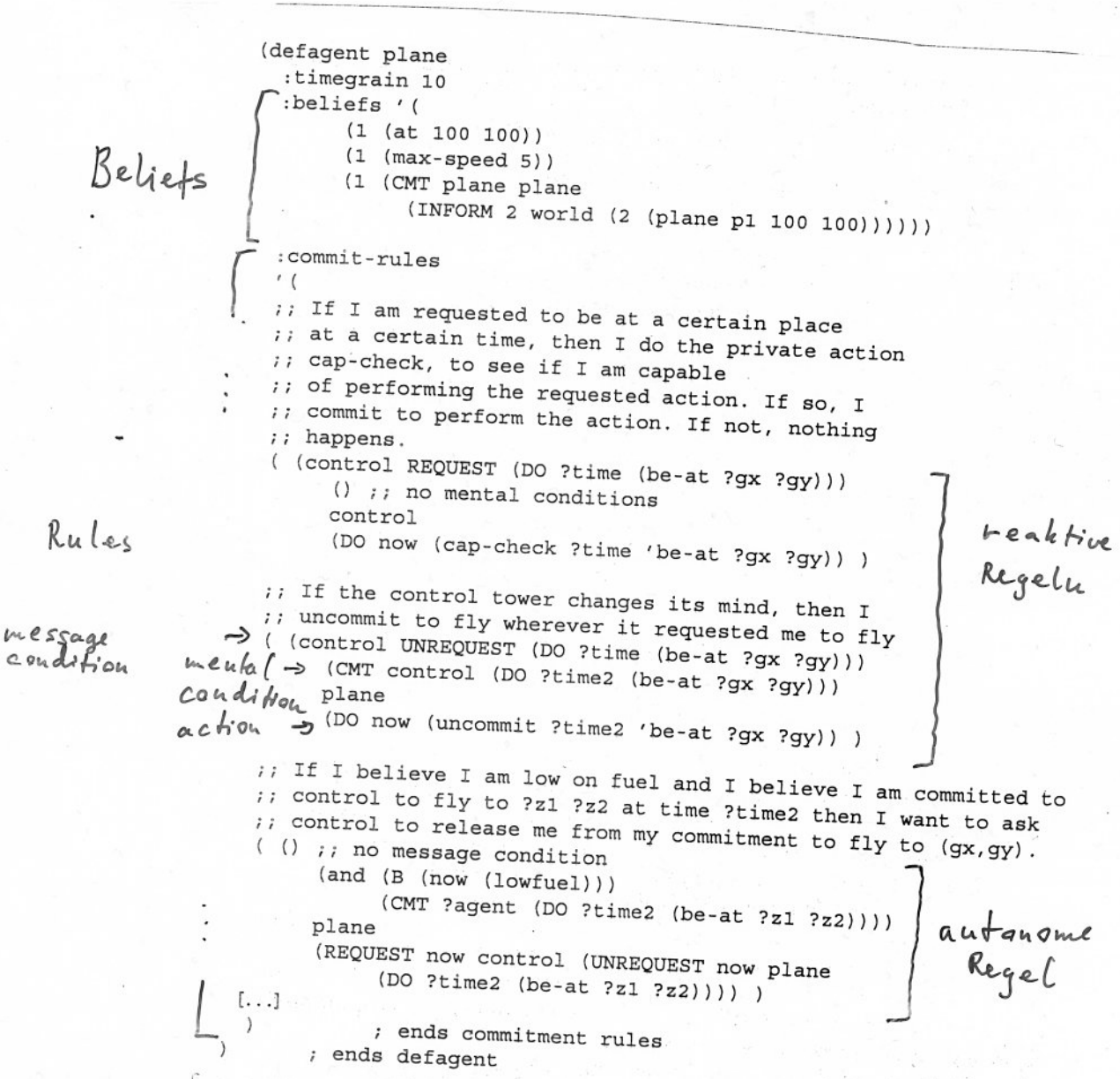
environment propositions: Akzeptierter Input

component propositions: Möglicher Output

Beispiel: 'stack' agent.

Interface: stack (pop, push) [popped, full]

Id	environment propositions	component propositions




```
(defagent plane
  :timegrain 10
  :beliefs '(
       (1 (at 100 100))
       (1 (max-speed 5))
       (1 (CMT plane plane
            (INFORM 2 world (2 (plane p1 100 100))))))

  :commit-rules
  '(
  ;; If I am requested to be at a certain place
  ;; at a certain time, then I do the private action
  ;; cap-check, to see if I am capable
  ;; of performing the requested action. If so, I
  ;; commit to perform the action. If not, nothing
  ;; happens.
  ( (control REQUEST (DO ?time (be-at ?gx ?gy)))
       () ;; no mental conditions
       control
       (DO now (cap-check ?time 'be-at ?gx ?gy)) )

  ;; If the control tower changes its mind, then I
  ;; uncommit to fly wherever it requested me to fly
  ( (control UNREQUEST (DO ?time (be-at ?gx ?gy)))
       (CMT control (DO ?time2 (be-at ?gx ?gy)))
       plane
       (DO now (uncommit ?time2 'be-at ?gx ?gy)) )

  ;; If I believe I am low on fuel and I believe I am committed to
  ;; control to fly to ?z1 ?z2 at time ?time2 then I want to ask
  ;; control to release me from my commitment to fly to (gx,gy).
  ( () ;; no message condition
       (and (B (now (lowfuel)))
            (CMT ?agent (DO ?time2 (be-at ?z1 ?z2))))
       plane
       (REQUEST now control (UNREQUEST now plane
            (DO ?time2 (be-at ?z1 ?z2)))) )

  [...]
  )
       ; ends commitment rules
  ; ends defagent
```

Figure 3.4: Part of the definition of a 'plane' agent in AGENT0 [Torrance and Viola, 1991].

In (x, y) Dirt (x, y) Facing (d)

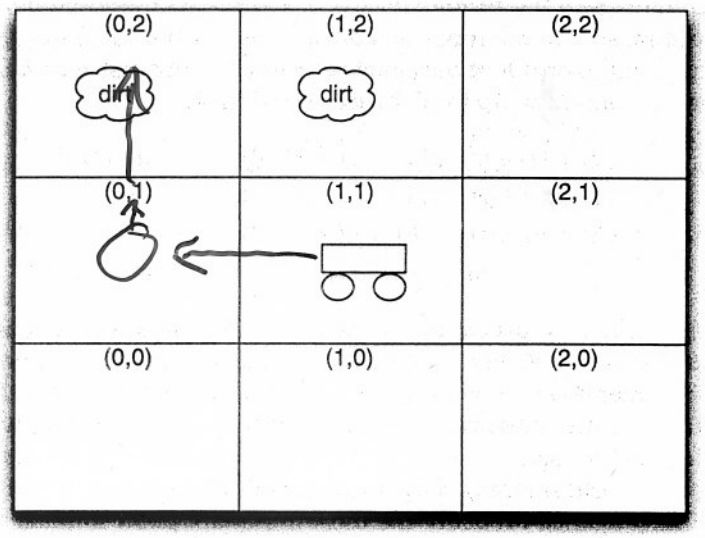


Figure 3.3: Vacuum world.

Staubsaugeragent

- Sensor (Dreck)
- Rechts drehen (90°)
- move 1 step (horizontale / vertikale)

Wissen

- Wo bin ich?
- Blickrichtung (Norden / Osten / Süden / Westen)

Goal

- Kontinuierliche Raumsäuberung

Aktionen

- forward
- suck [In (x, y) ∧ Dirt (x, y) → Do (suck)]
- turn

new: D × Per → D

next (DB, P) = (DB \ old (DB)) ∪ new (DB, P)

Gruppe 1

↳ Deliberative Agenten haben eine Menge von prädikatenlogischen Formeln als Wissensbasis.

| Mädchen (Laura)
| Interessant (this_seminar)

Diese funktionieren (wie beim Menschen) als „Beliefs" und können wahr oder falsch sein.

↳ DB = Menge von präd. Formeln (z.B: Aktueller Zustand) | $DB \subseteq D$

D = Komplette Wissensbasis

↳
- see: $S \rightarrow Per$
- next: $D \times Per \rightarrow D$
- action: $D \rightarrow Ac$


```
1.  function action(DB : D) returns an action Ac
2.  begin
3.      for each α ∈ Ac do
4.          if DB ⊢ρ Do(α) then
5.              return α
6.          end-if
7.      end-for
8.      for each α ∈ Ac do
9.          if DB ⊬ρ ¬Do(α) then
10.             return α
11.         end-if
12.     end-for
13.     return null
14. end function action
```

Begründete Aktionen

Aktionen die nicht Verboten sind

Figure 3.2: Action selection as theorem proving.

=> Verhalten bestimmt aus Wissensbasis und Ableitungsregeln