Statecharts: A Visual Formalism for Complex Systems:

David Harel
(communicated by A. Pnueli)
1986

https://www.inf.ed.ac.uk/teaching/courses/seoc/2005_2006/resources/statecharts.pdf

About me:

EE (PEng) 8T1
Also, studied in Core Physics (7T9)

Compilers, OSs, DSLs, embedded systems.

Ran s/w consultancy 25+ years

I first read Harel's paper in 1987, then applied it to
Injection Molding machines project, to replace PLCs.

Current Interests:
Diagrams-as-Syntax
Expression of design intent,
Software Dev —> Engineering + guarantees.

STATECHARTS:
A VISUL FORMALISM FOR COMPLEX SYSTEMS

- 44 pages
- 49 figures
- Hierarchy
- Concurrency
- Communication
- Structured control flow
- 9 sections (meat of notation in sections 2-5)

The notation was used originally for avionics
(closed source).

This paper describes a Citizen Digital Watch
as its demo.

The Digital Watch is reverse-engineered, and
the diagrams indicate that the watch was
“designed by committee”

1. Introduction

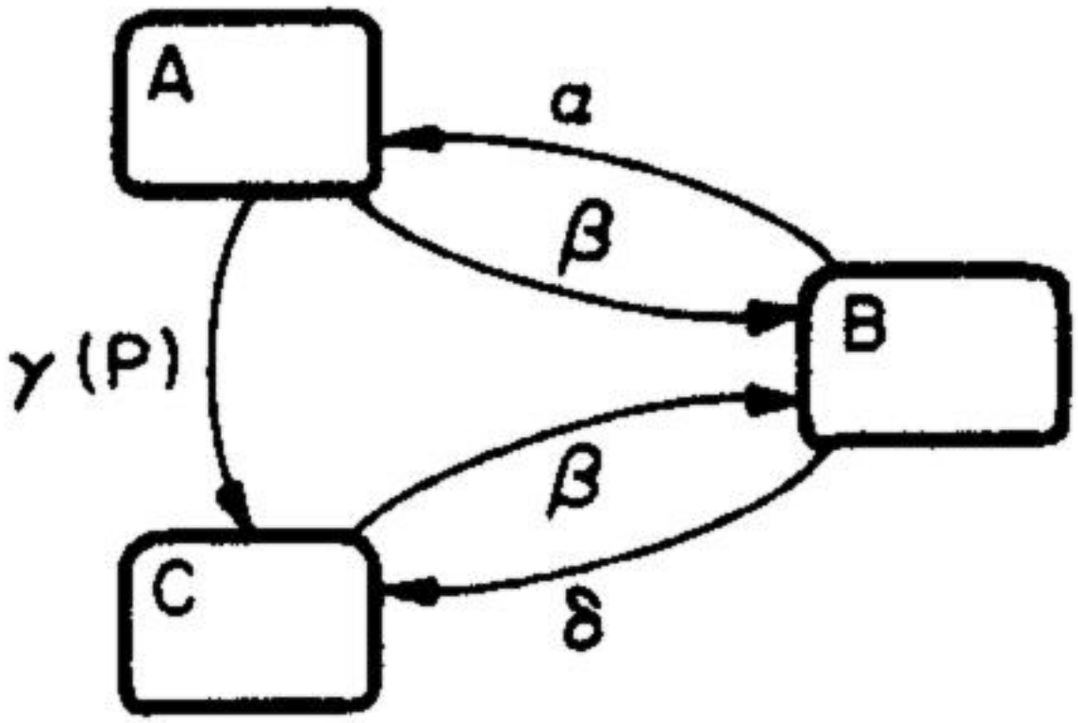

Fig. 1.

Simple State Diagram

A/B/C are States

alpha/beta/delta/gamma are Events

P is a guard predicate

2. State-levels: Clustering and Refinement

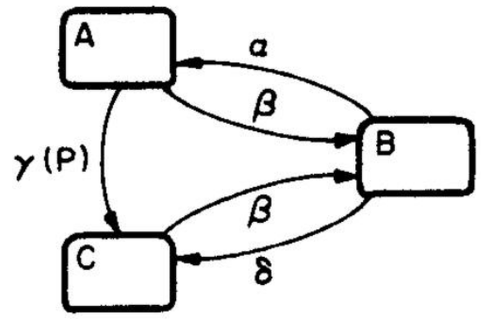

Fig. 1.

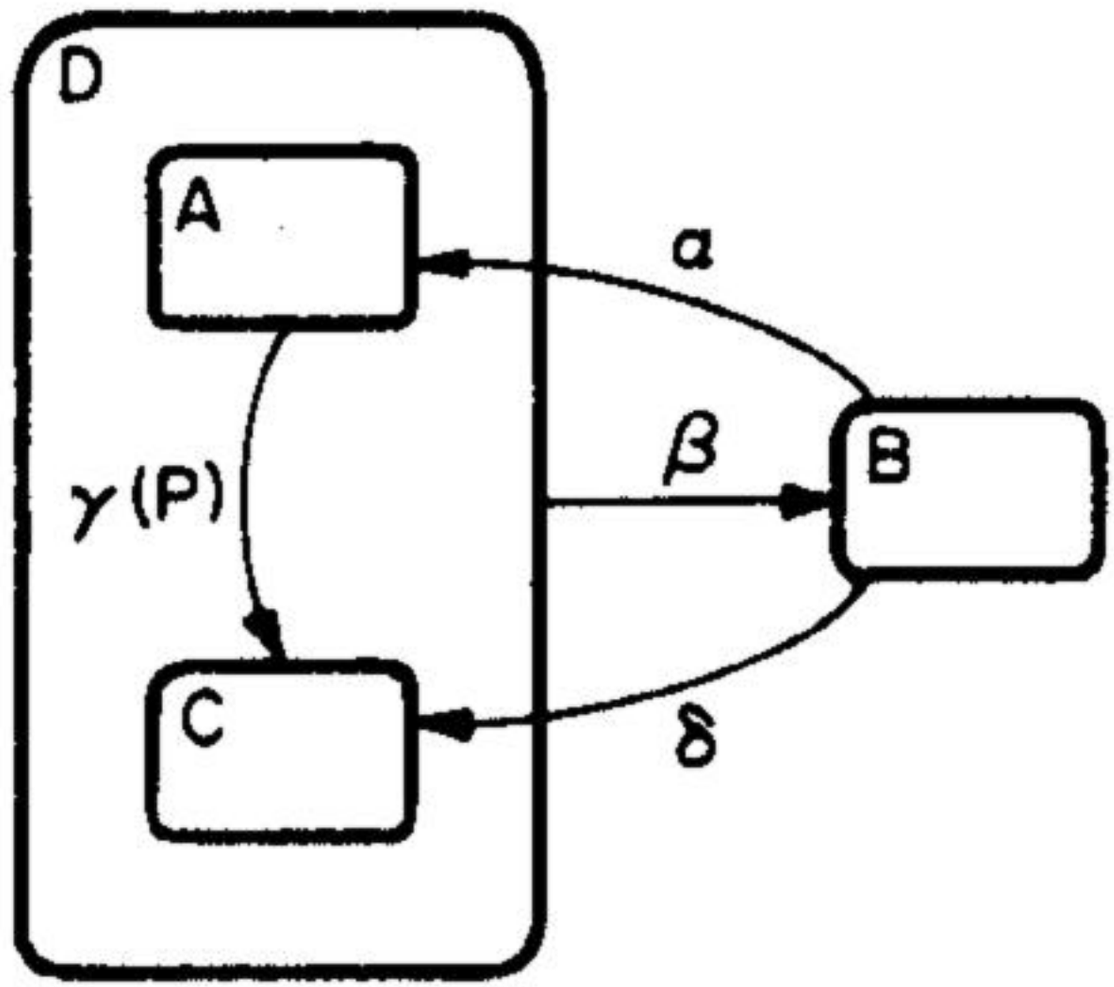

Fig. 2.

Clustering

A & C moved inside D

All *beta* transitions combined into a single transition

Children of D (A/C) cannot override parent's *beta* transition

(opposite of inheritance)

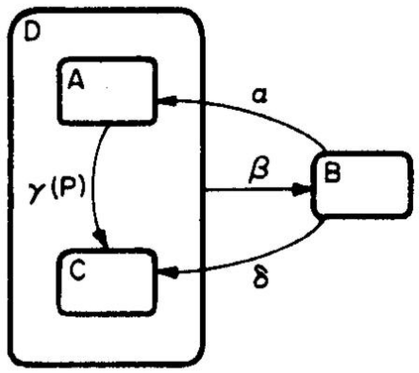

Fig. 2.

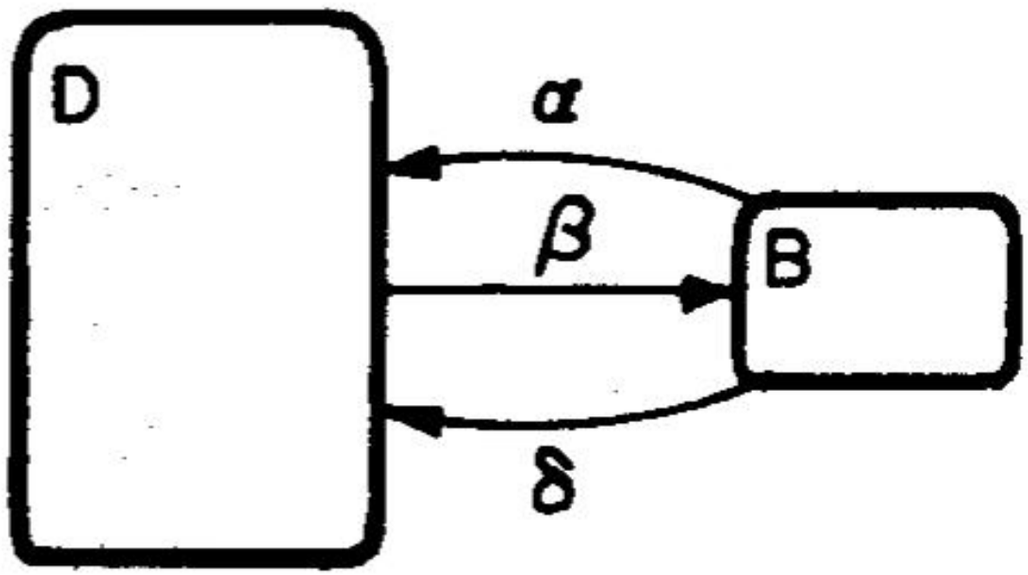

Fig. 3.

Different views
of same states

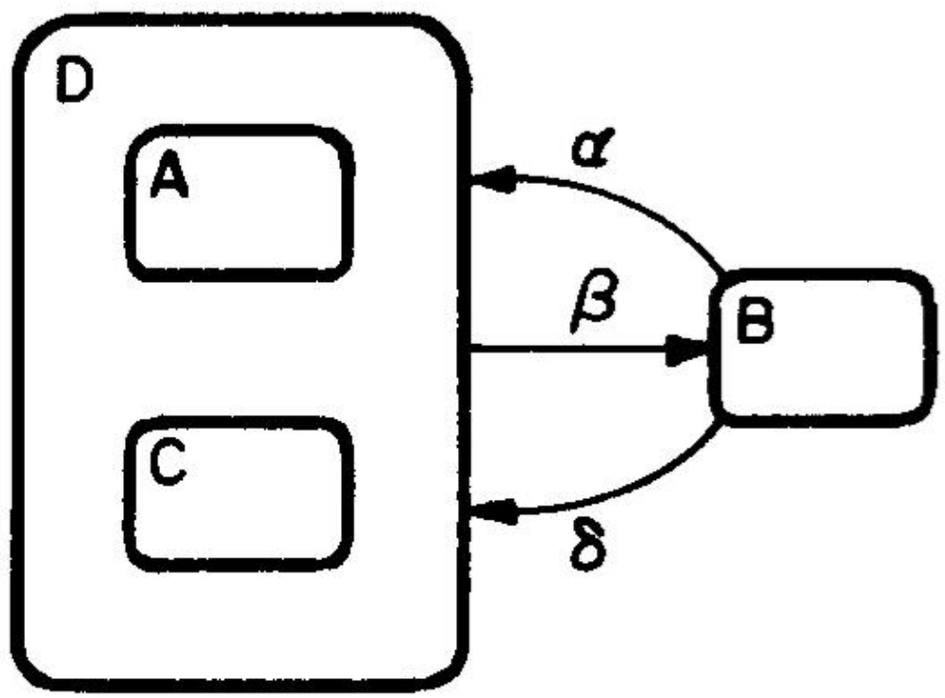

Fig. 4.

Skip

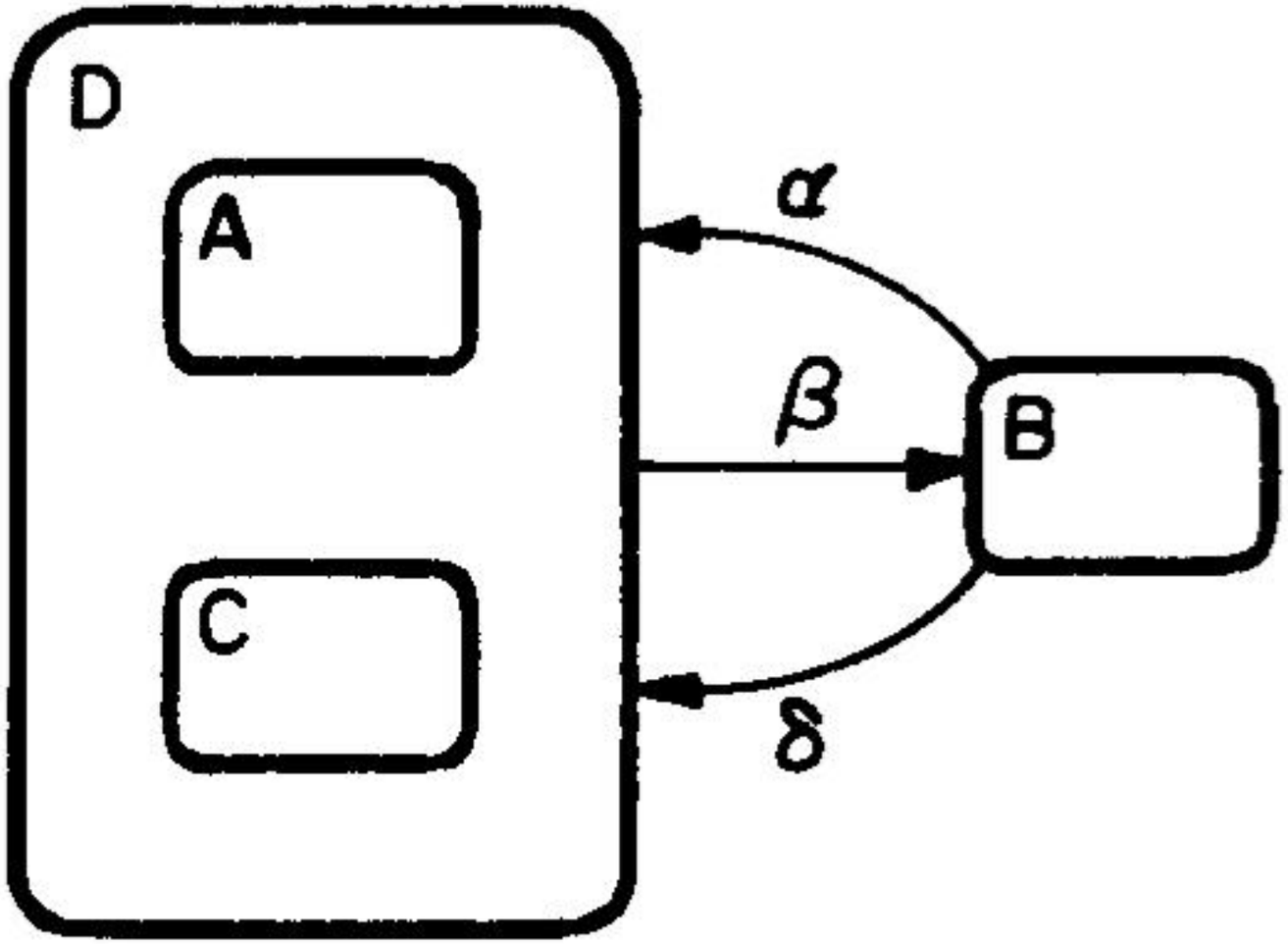

Fig. 4.

- Running example (Citizen Quartz Multi-Alarm III watch)

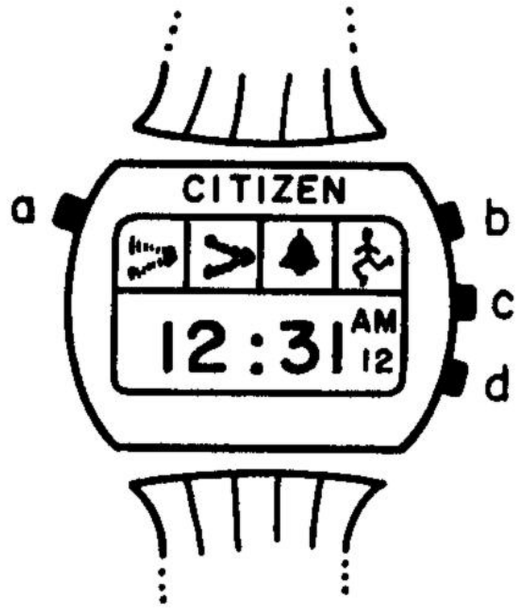

Fig. 7.

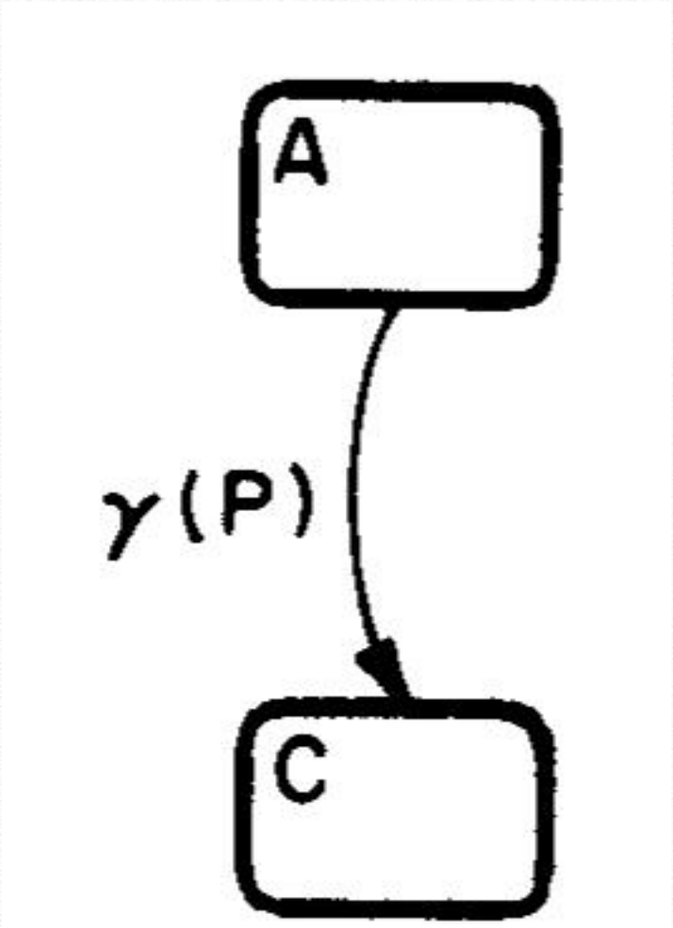

Fig. 5.

Skip

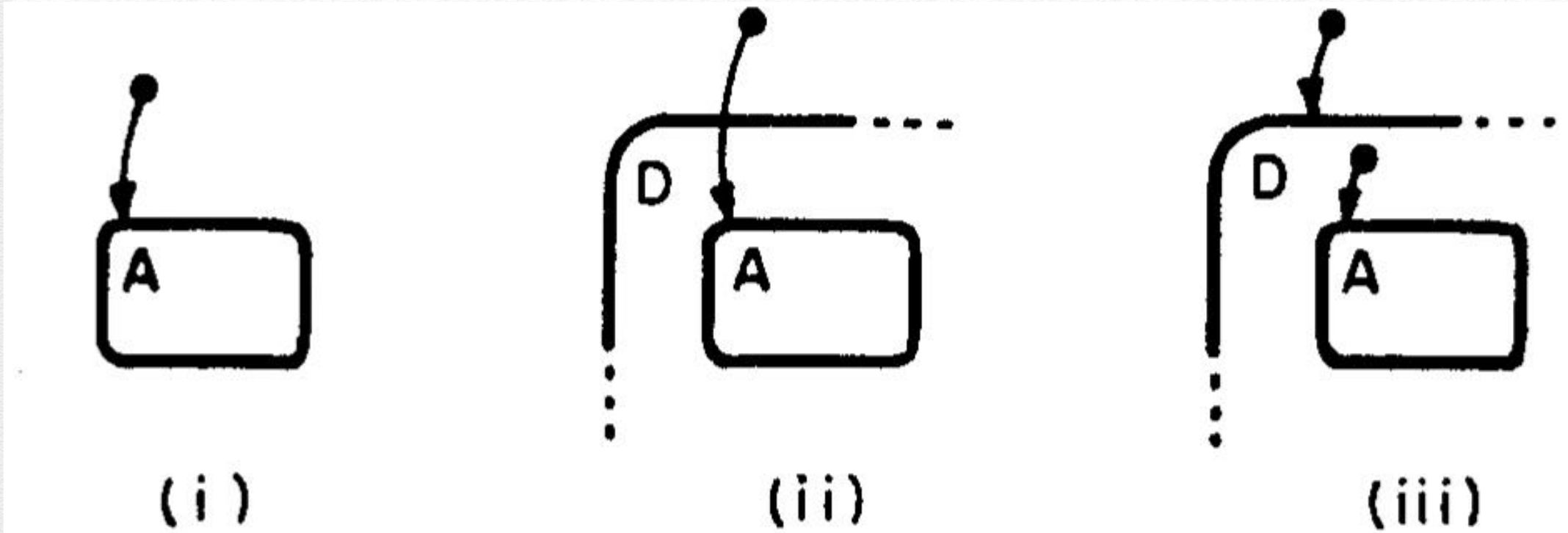

Fig. 6.

Default Entry Point

(i) Enter A by default
(ii) Enter D.A by default
(iii) Enter D by default, then Enter A by default (in D)

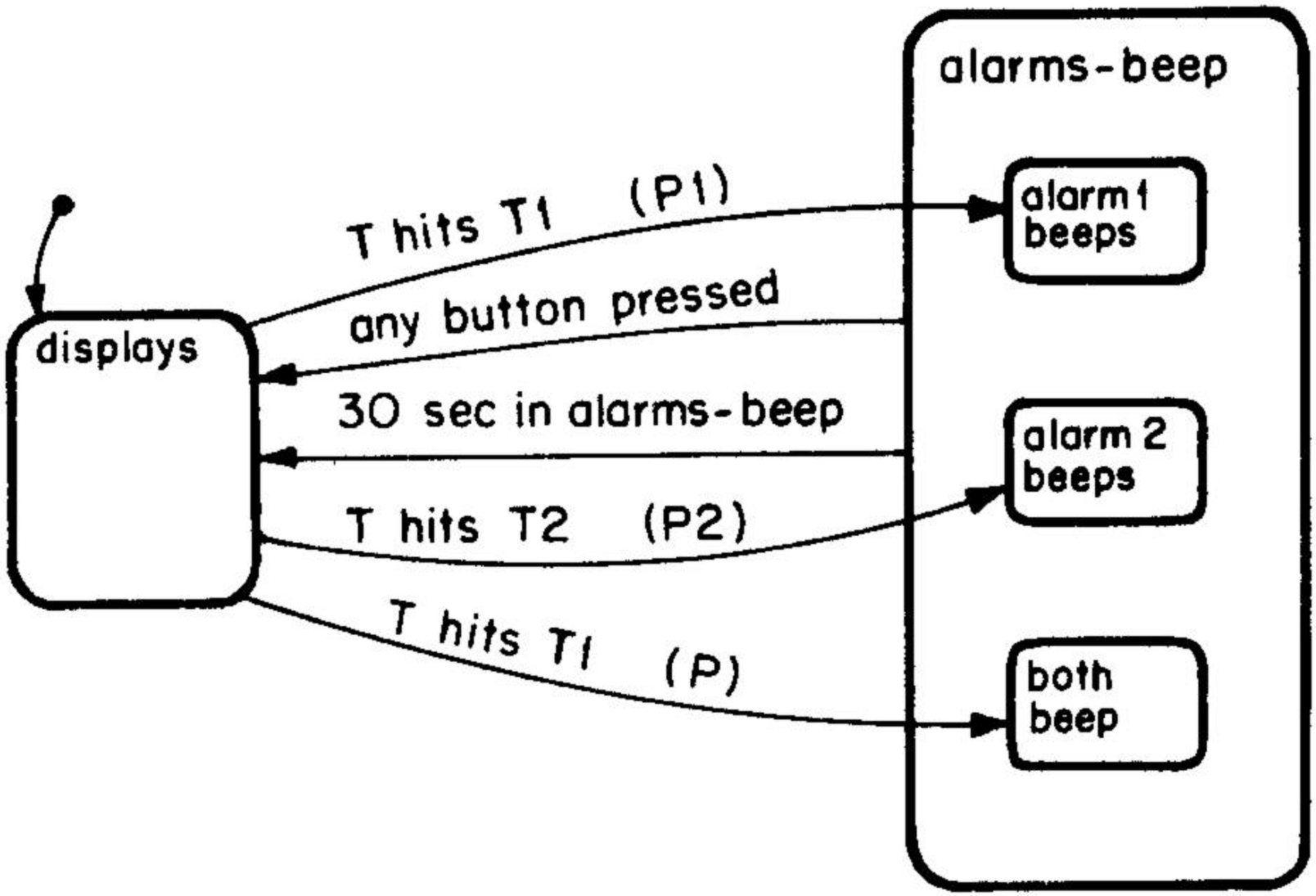

Fig. 8.

State Explosion

"any button pressed" is 3 arrows
"30 sec in alarms-beep" is 3 arrows
Both compressed to 1 arrow (each) through clustering.

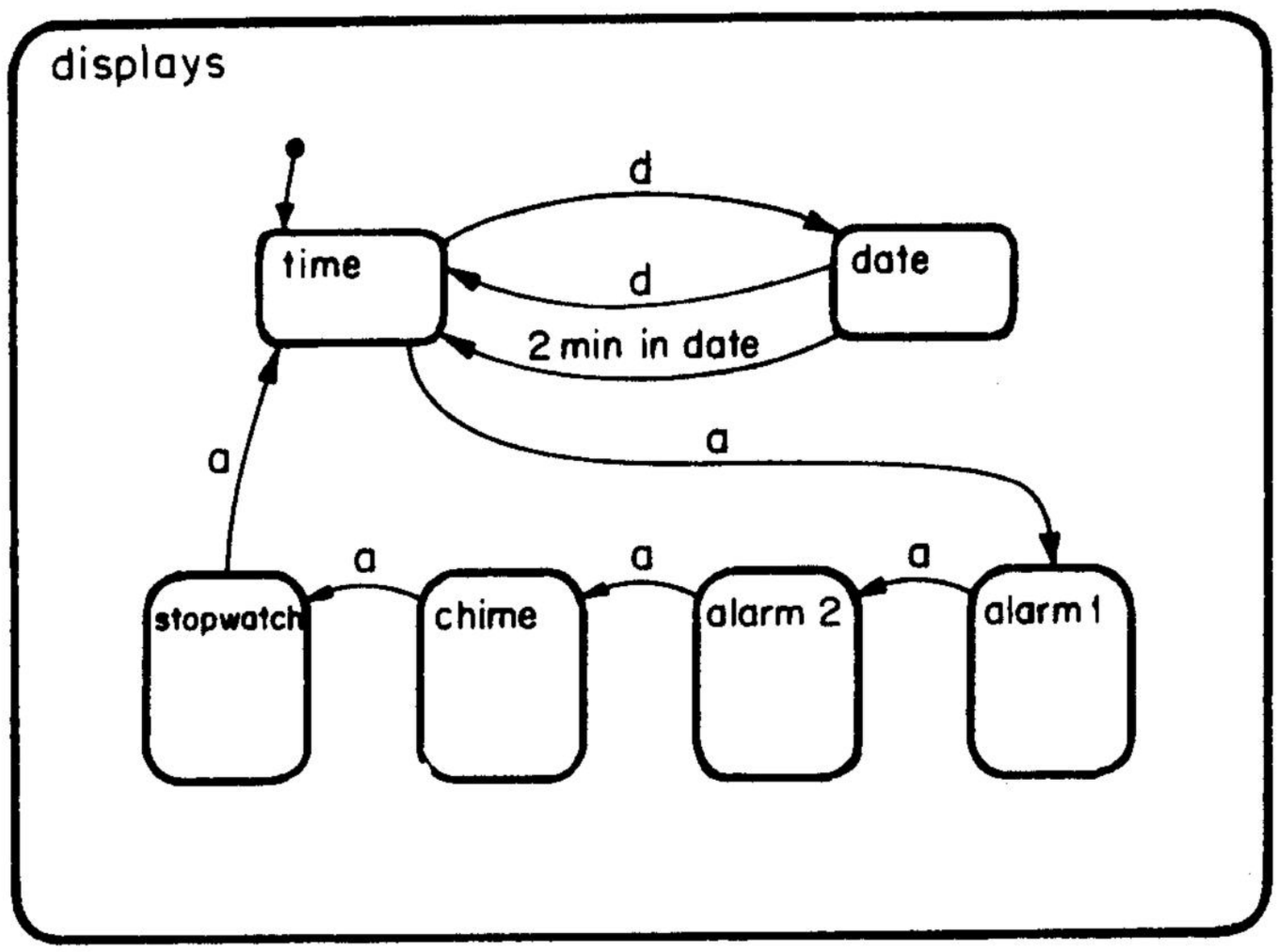

Fig. 9.

Enter ‘time’ by default
When in ‘time’ {
  if “d” is pressed, goto ‘date’
  if “a” is pressed, goto ‘alarm1’
}
When in ‘alarm1’, 4 more “a” presses will goto ‘time’
When in ‘date’, 1 more “d” press, or 2 minutes, will goto ‘time’

Skip

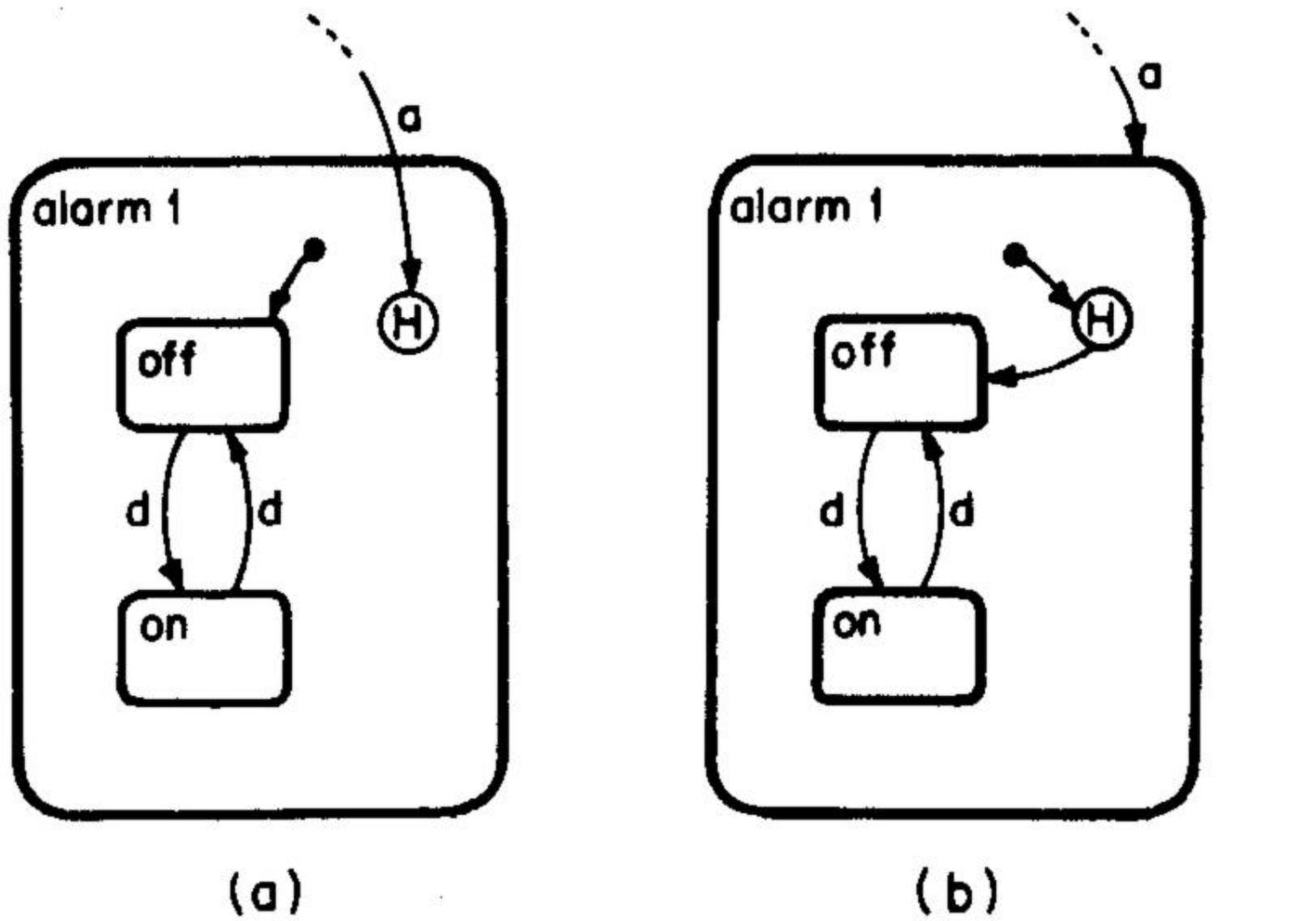

Fig. 10.

History

Enter default (off) if first time
Else enter previous state
Both diagrams have same result

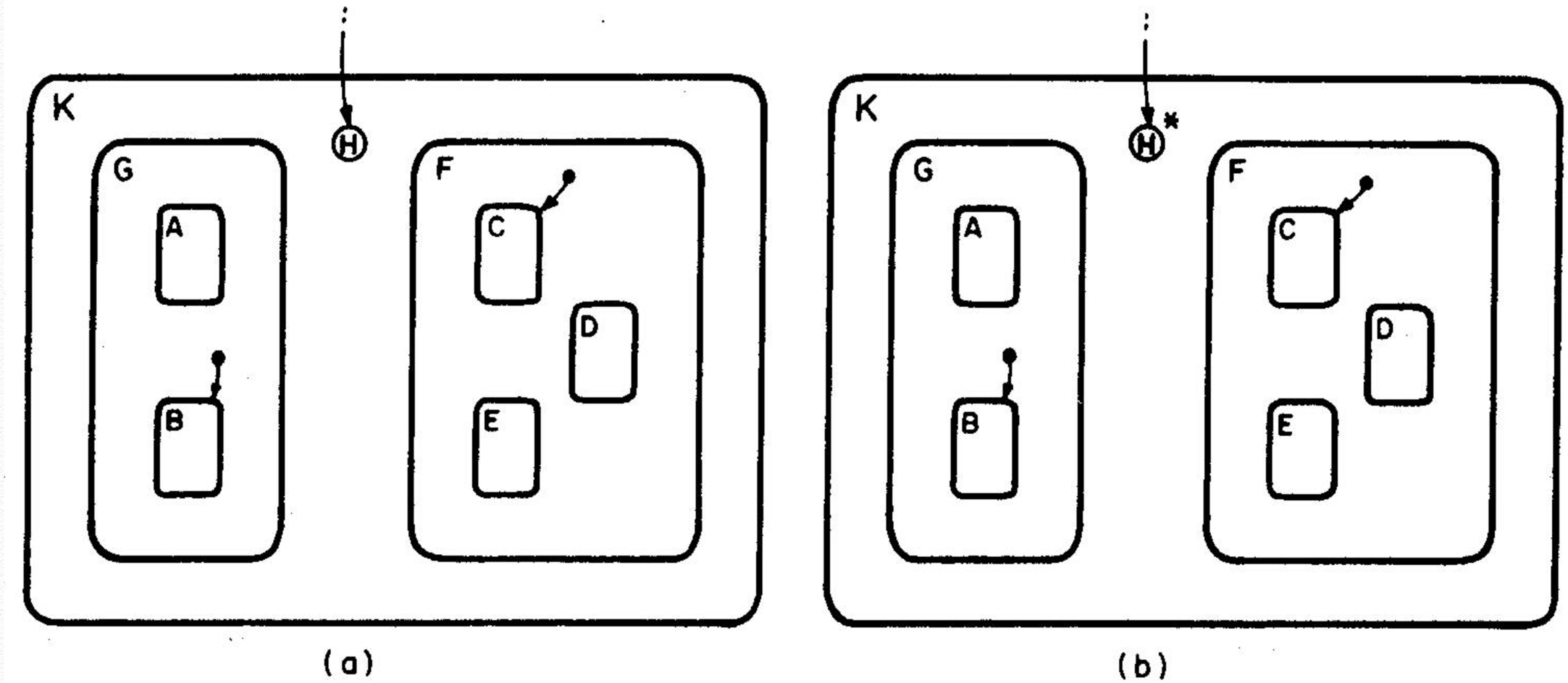

Fig. 11.

History

(a) 1-level "history" chooses K.G or K.F (i.e. K.G.B or K.F.C)

(b) "deep history" uses most recent states
(K.G.A or K.G.B or K.F.C or K.F.D or K.F.E)

Skip

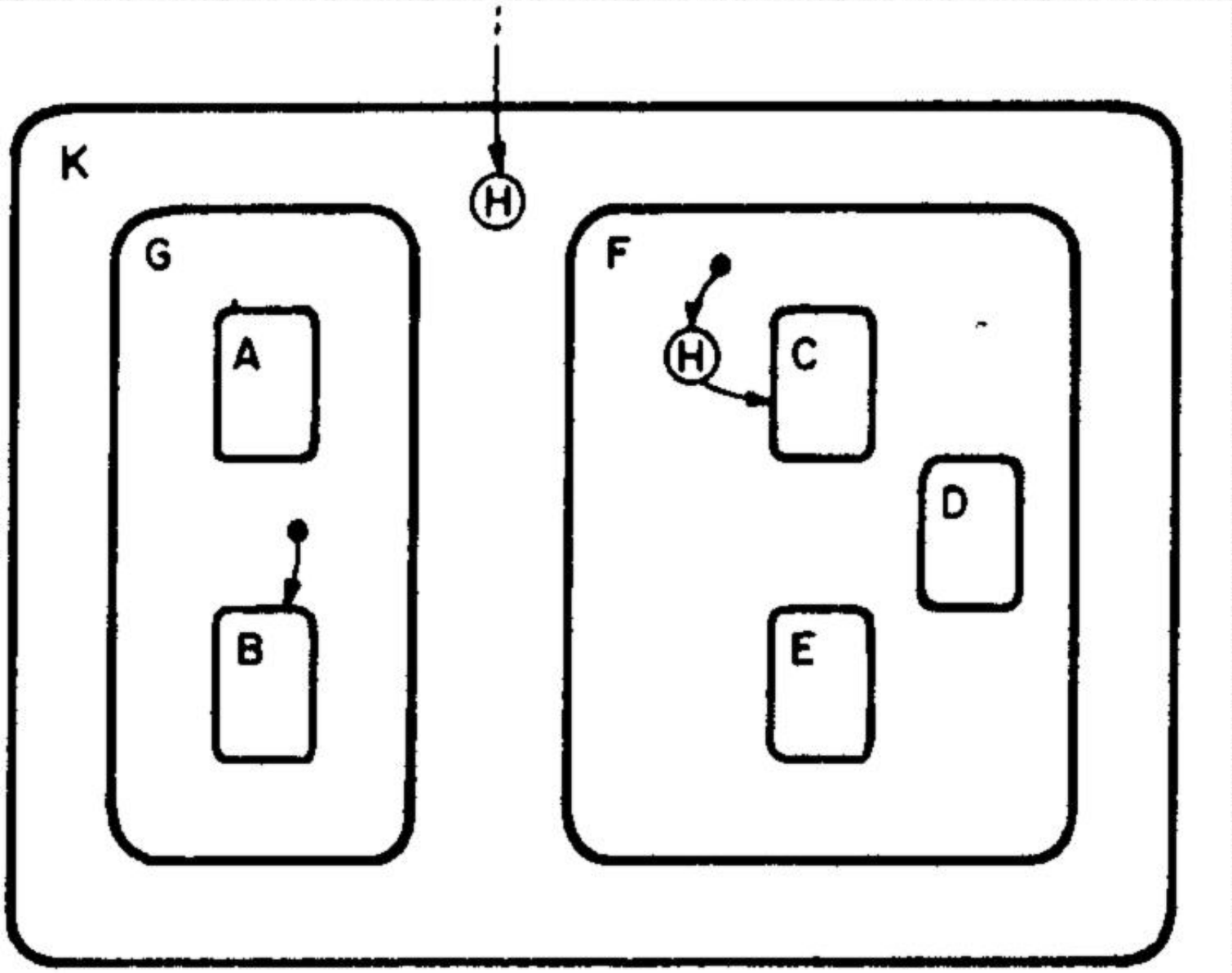

Fig. 12.

History

Enter K.G.B or K.F.C or K.F.D or K.F.E

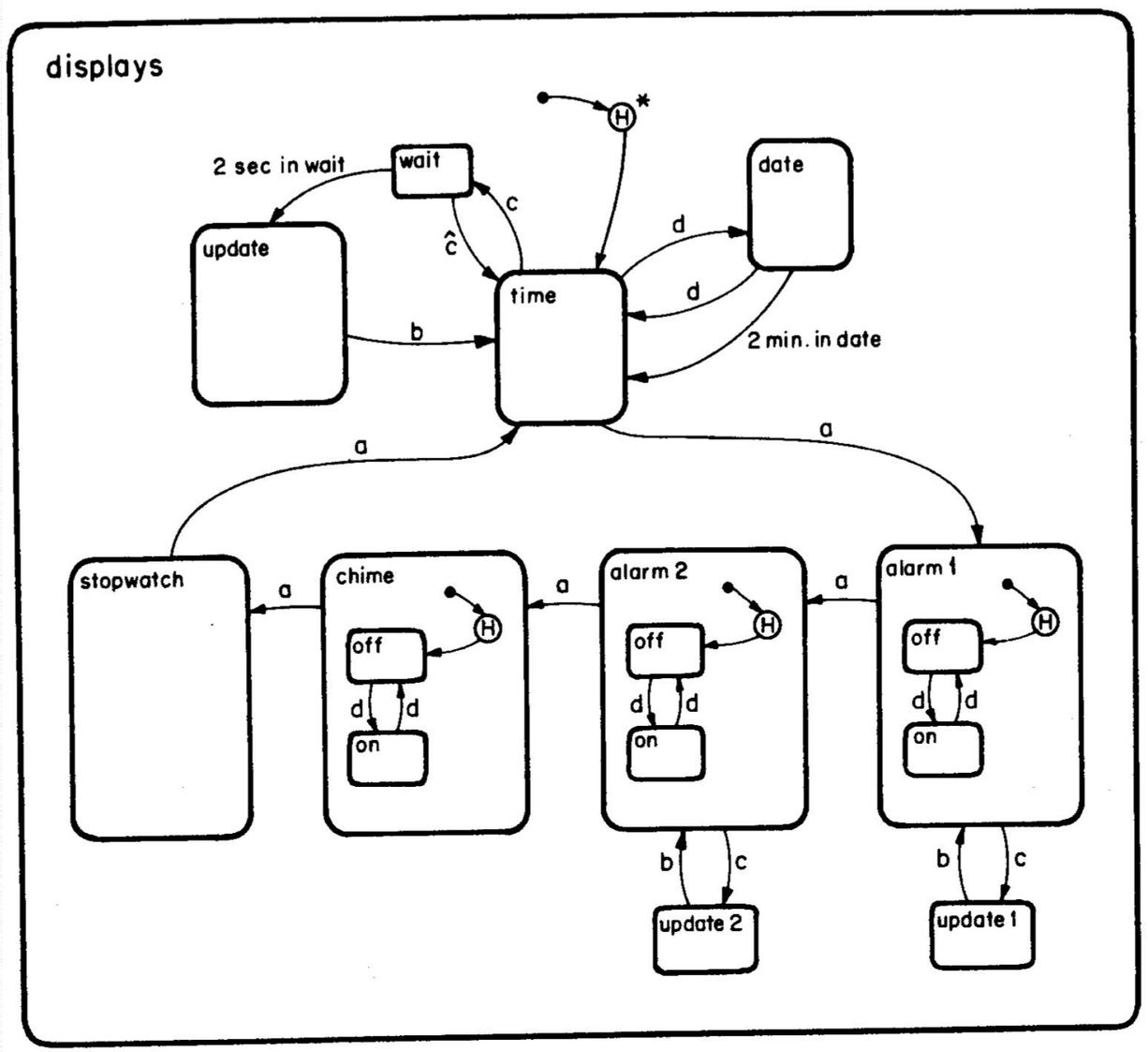

Fig. 13.

Time Delay

time —>on c down—> wait
wait —>on c up—>time
wait —>on 2 sec—>update

Underspecified?
c can be held down during update
can b be pressed while c down?
Edge-driven or value driven?
“c PUSHED down” vs. “c IS down”.
Is c-up ignored in ‘update’ / ‘time’?
(see semantics paper)

Observation:
Diagrams make some
semantic questions
easier to spot.

Skip

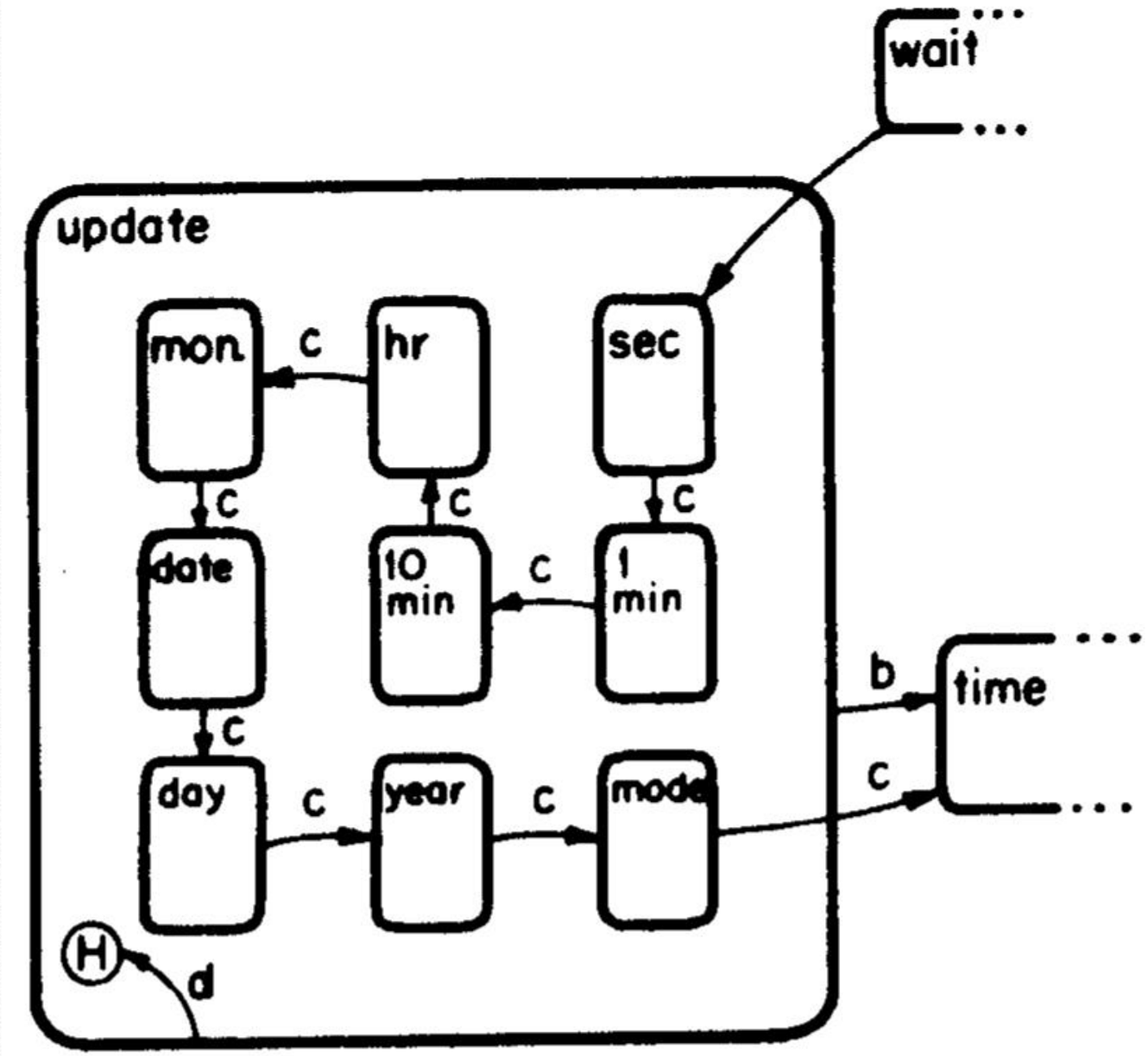

Fig. 14.

‘update’ from previous slide with more detail
Answer: c-up is ignored,
c-down is used to move
between substates

Skip

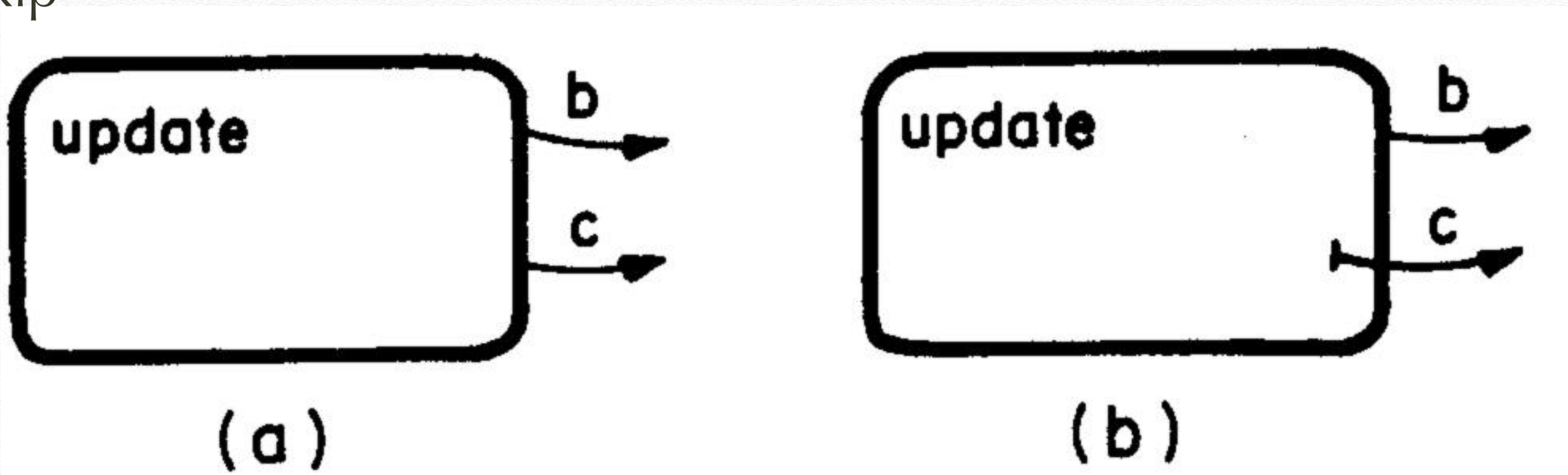

Fig. 16.

State Exits

(a) ‘update’ will exit if “b” or “c” occurs

(b) ‘update’ will exit if “b” occurs and, given appropriate condition, if “c” occurs

Skip

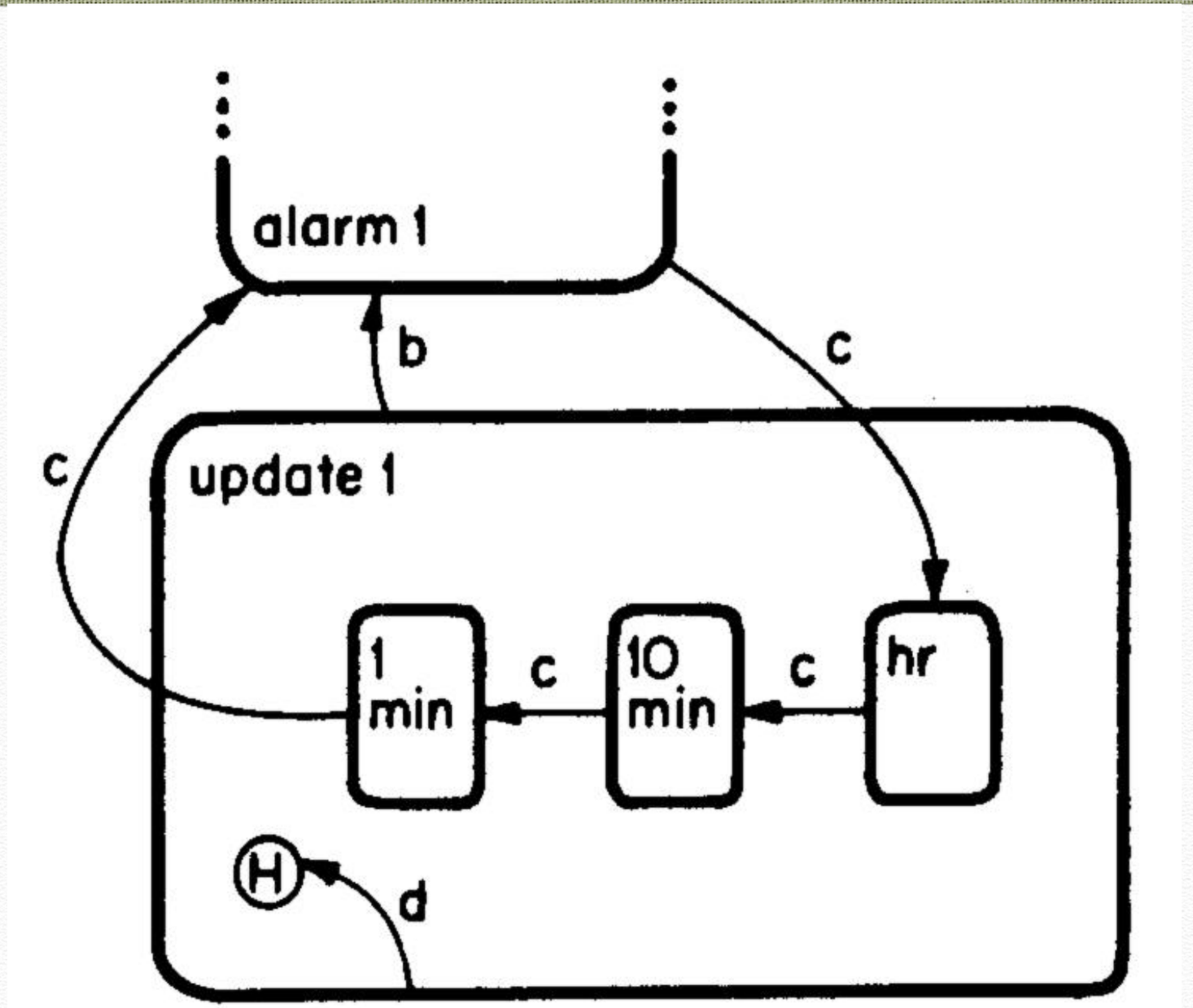

Fig. 15.

‘update1’ exits on c iff in state 1min
and always exits on b.
(see previous slide fig. 16)

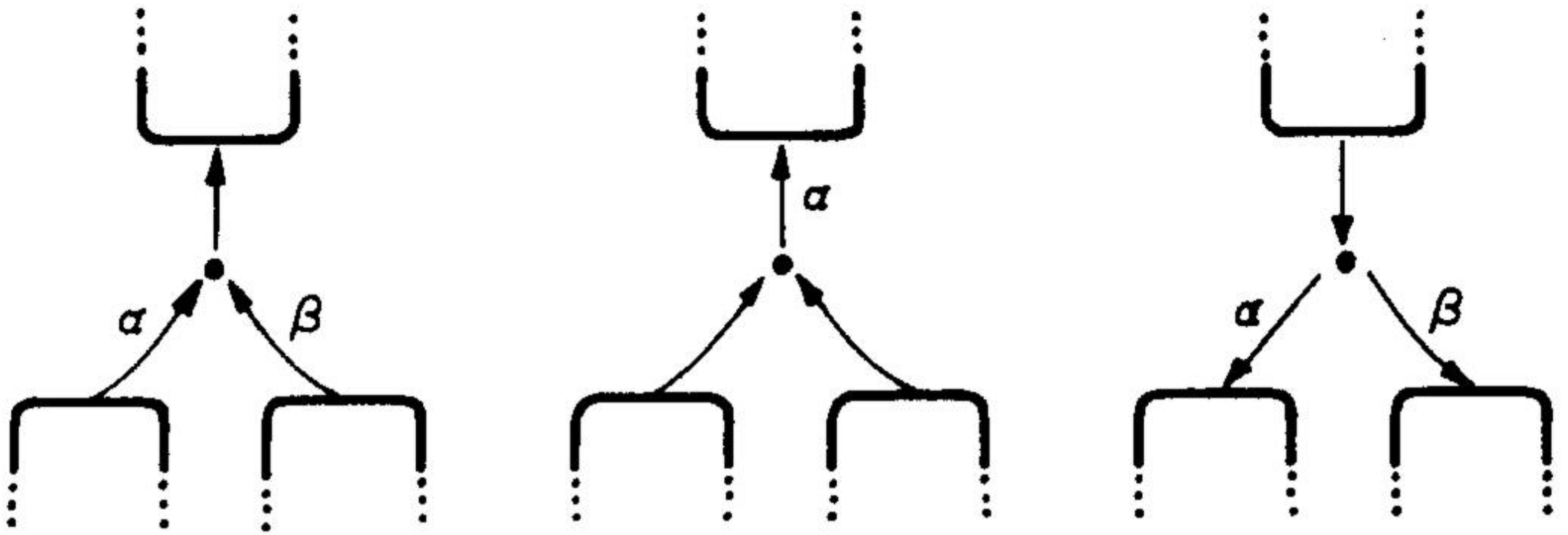

Fig. 17.

Economical Representation

Paper states that (c) is a contradiction

((b) with arrows reversed is a contradiction)

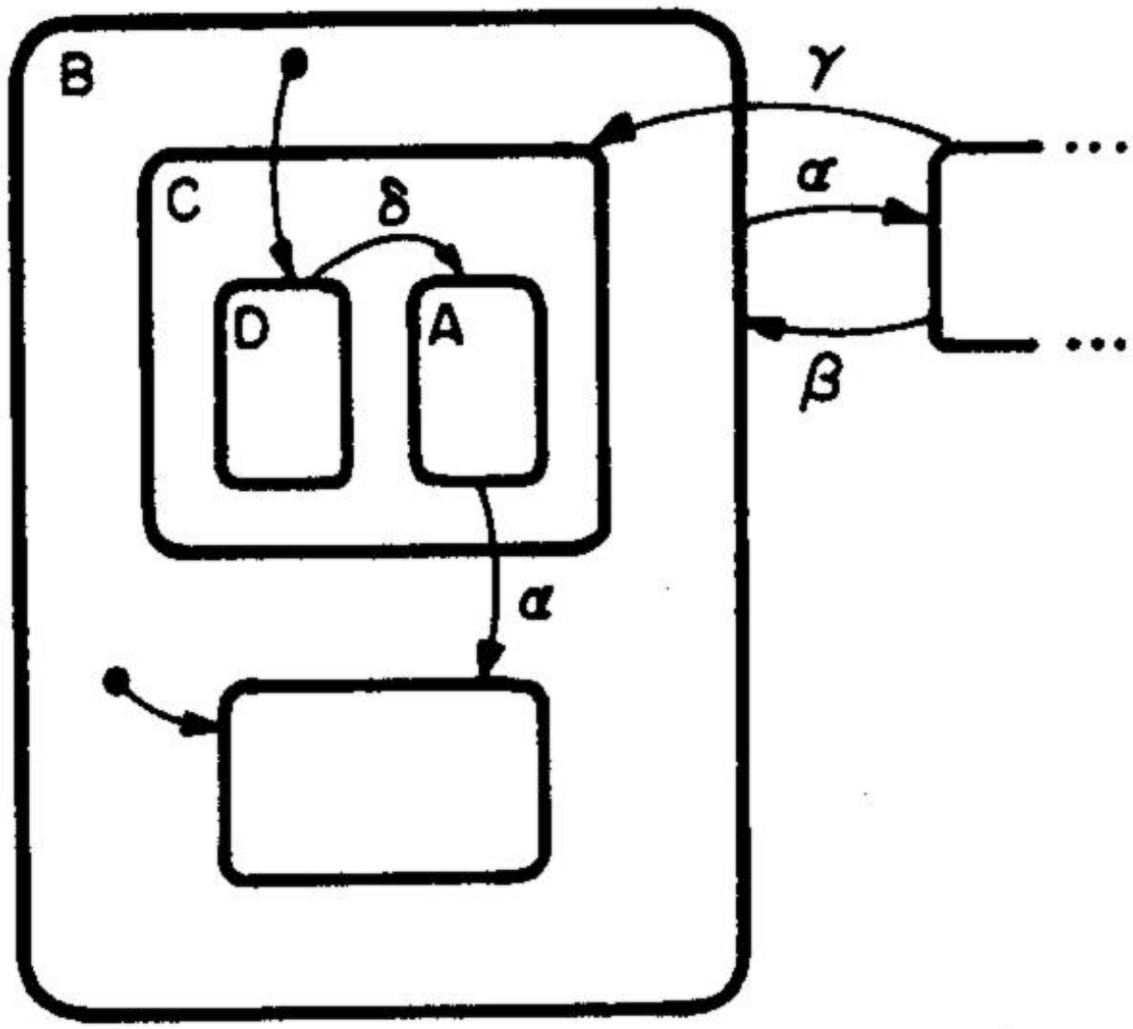

Fig. 18

Two Contradictions

1. Exit A on event *alpha*
2. Enter B on *beta*

C is underspecified (no default)

3. Orthogonality: Independence and concurrency

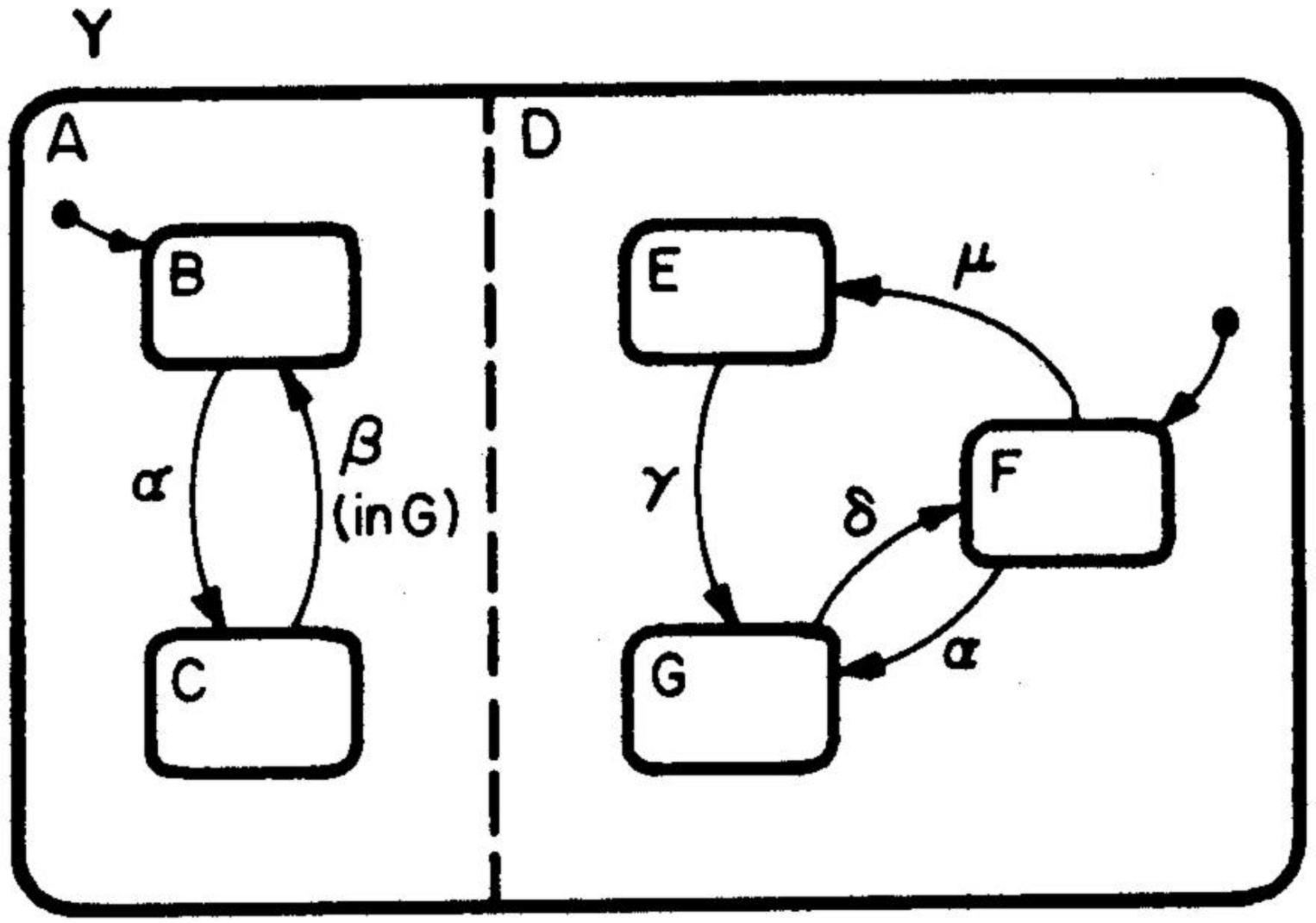

Fig. 19.

Two Simultaneous States

Default state is Y.A.B ^ Y.D.F
Transition from Y.A.C to Y.A.B guarded
by predicate “(in G)”

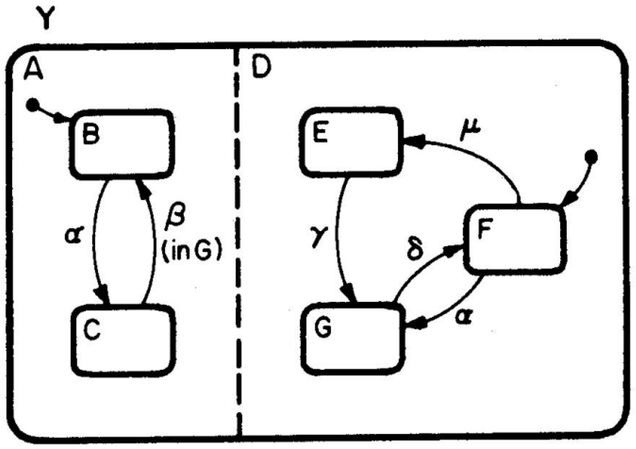

Fig. 19.

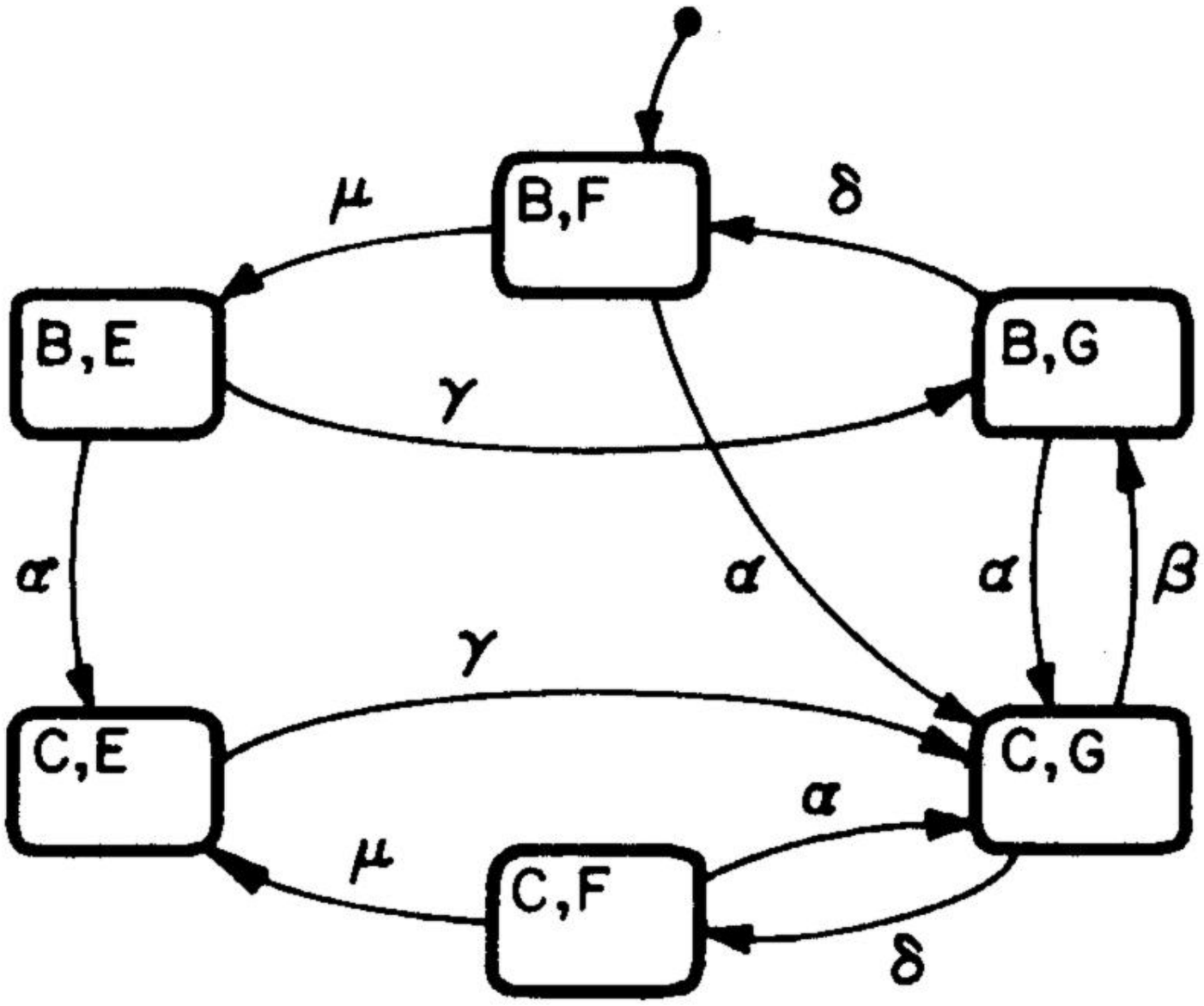

Fig. 20.

Fig. 20 is the AND-free equivalent of Fig. 19

Skip

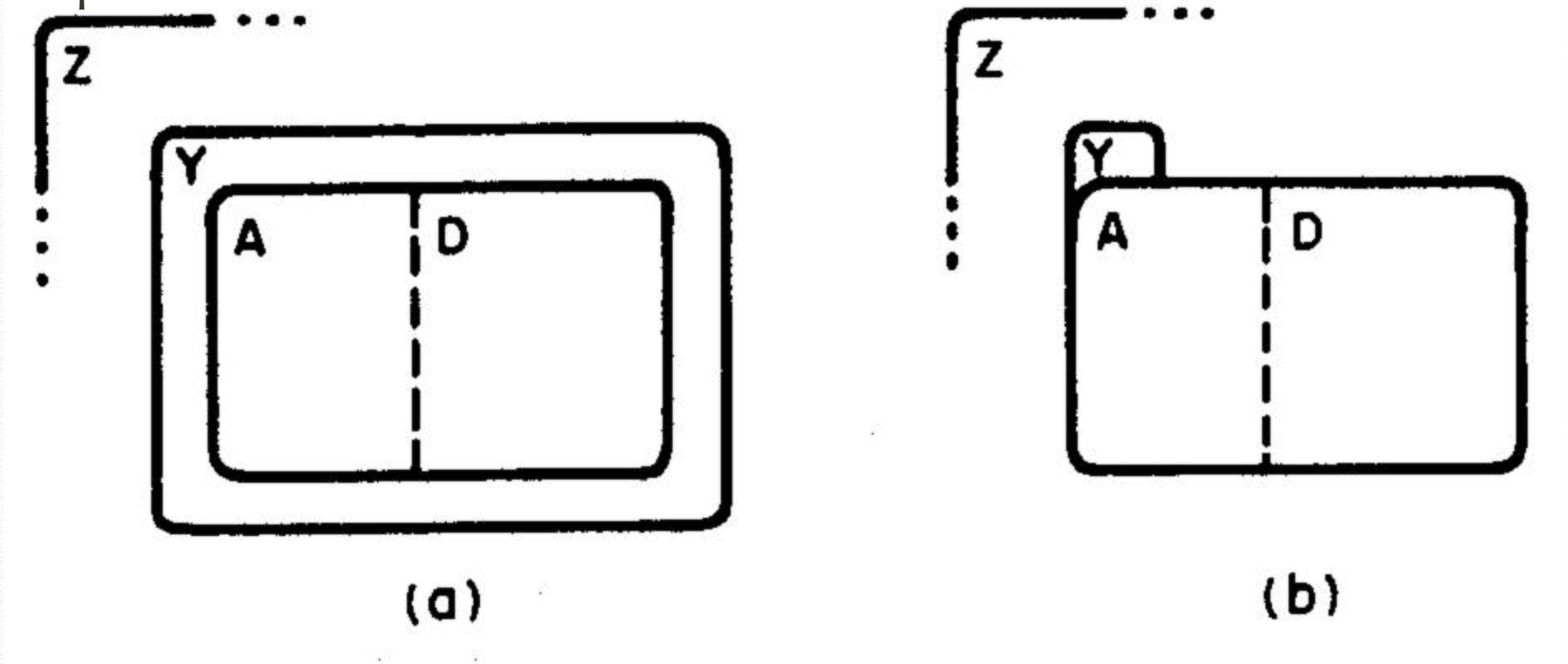

Fig. 21.

Two Ways to show that Y encompasses A & D

Skip

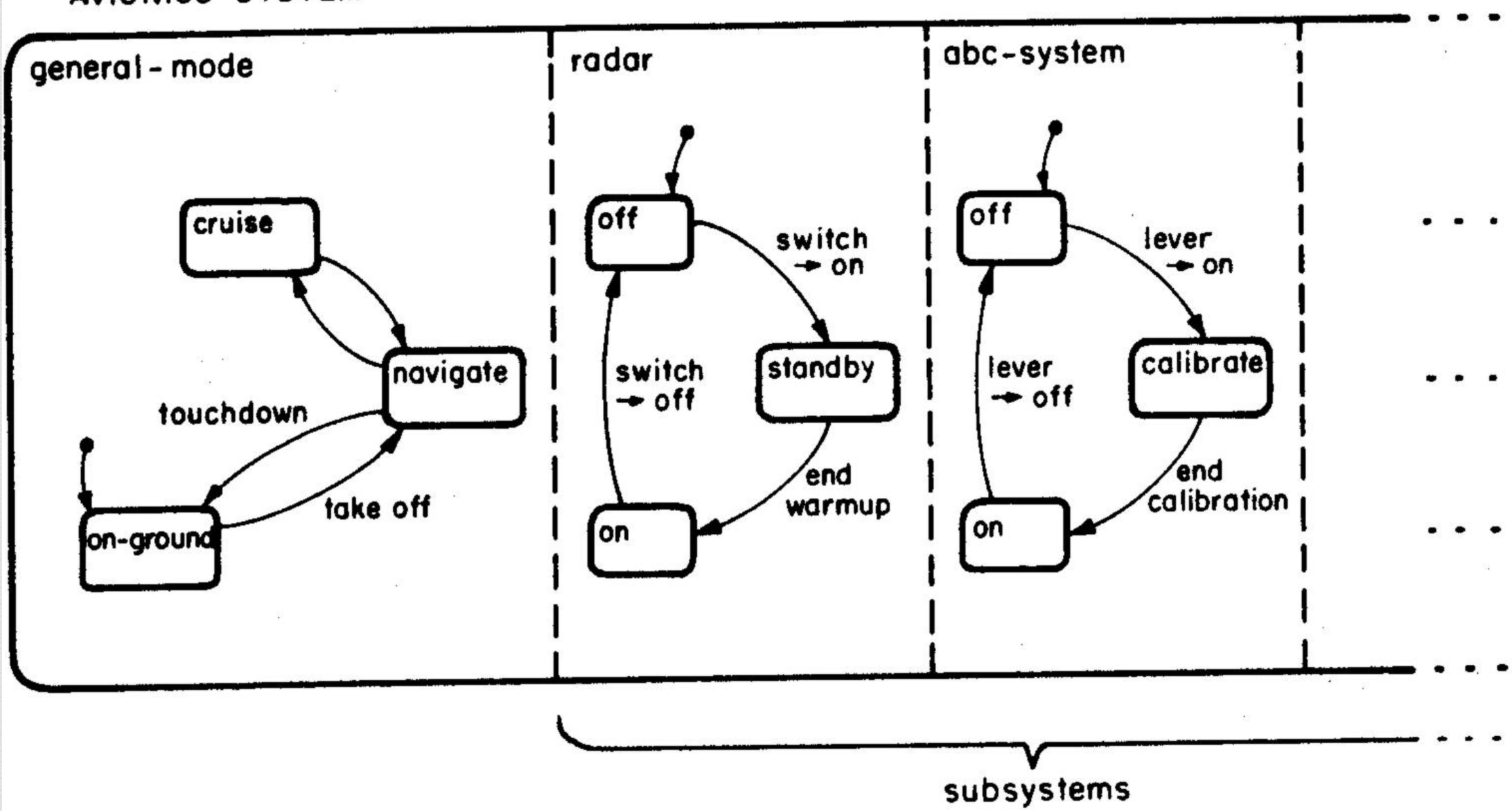

Fig. 22.

"An obvious application of orthogonality is in splitting a state in accordance with its physical subsystems."

Skip

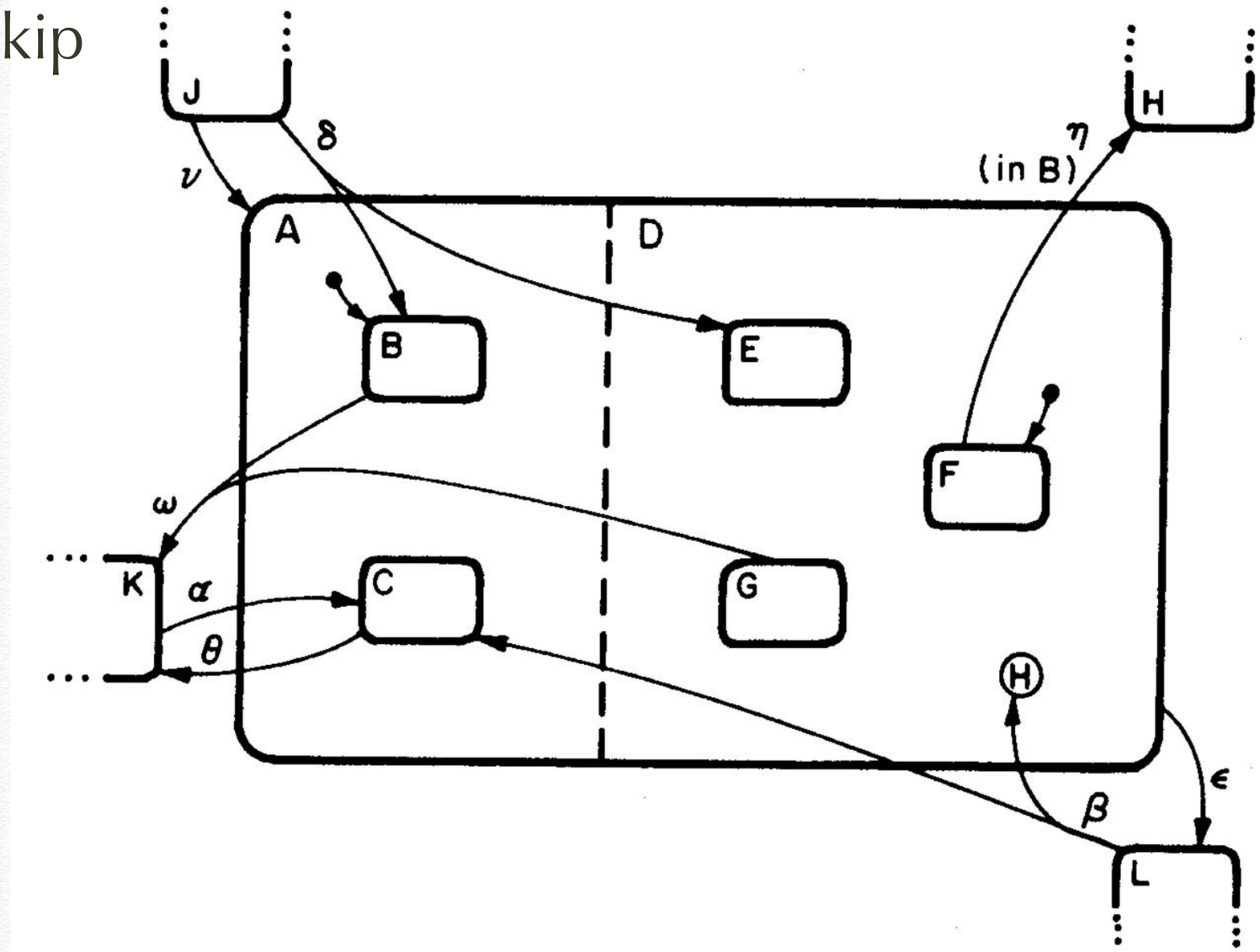

Fig. 23.

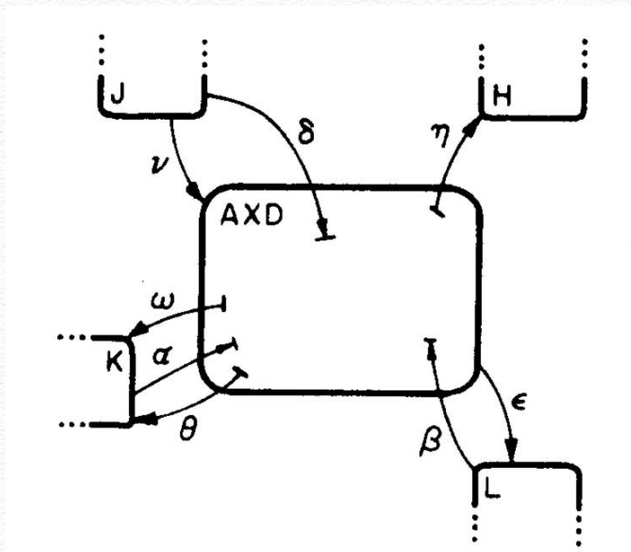

Fig. 24.

Entry and exit examples for orthogonal states.
Fig. 24 is a zoomed out version of Fig. 23,
N.B. “AxD” signifying concurrent states A and D

Skip

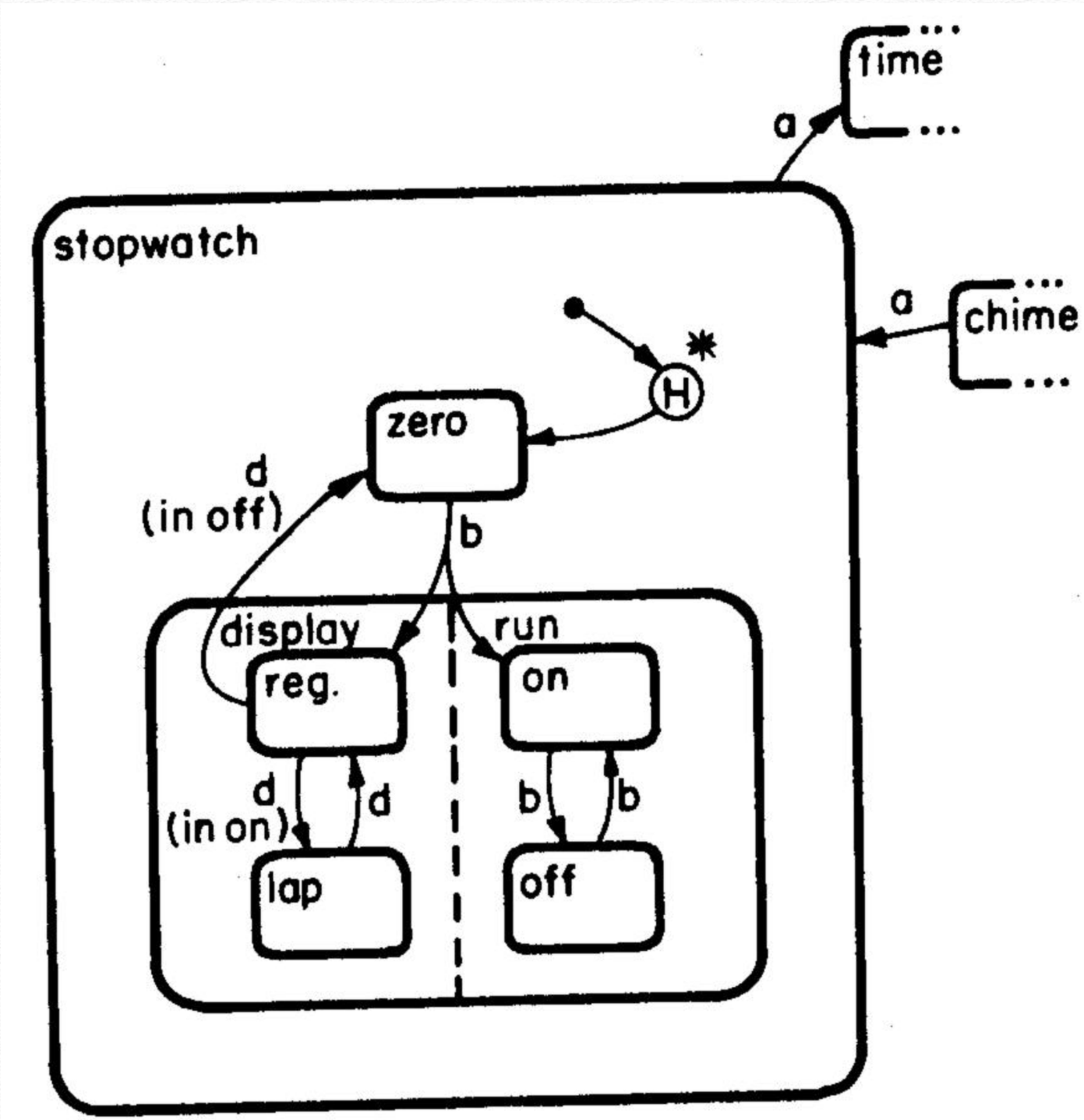

Fig. 25.

State ‘stopwatch’ zoomed in.

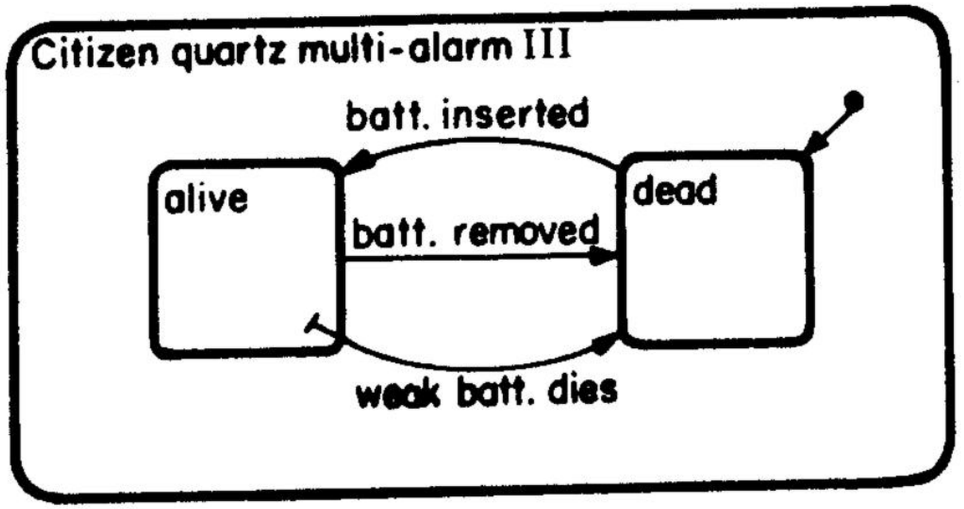

Fig. 26.

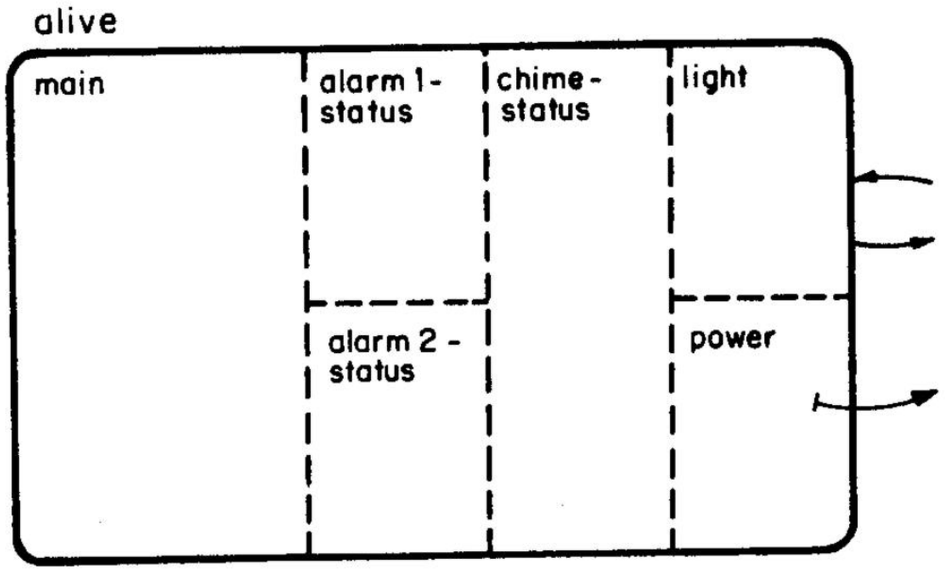

Fig. 27.

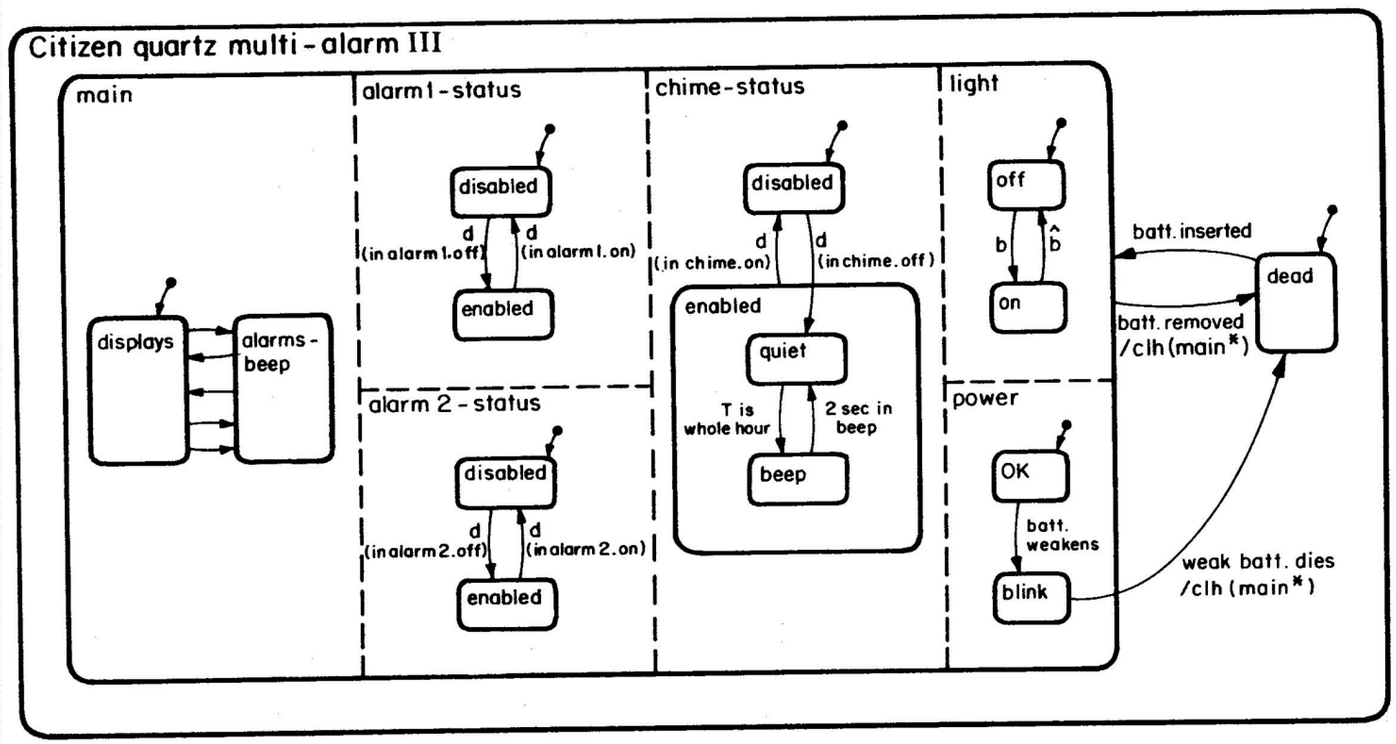

Fig. 28.

Top down specification of watch

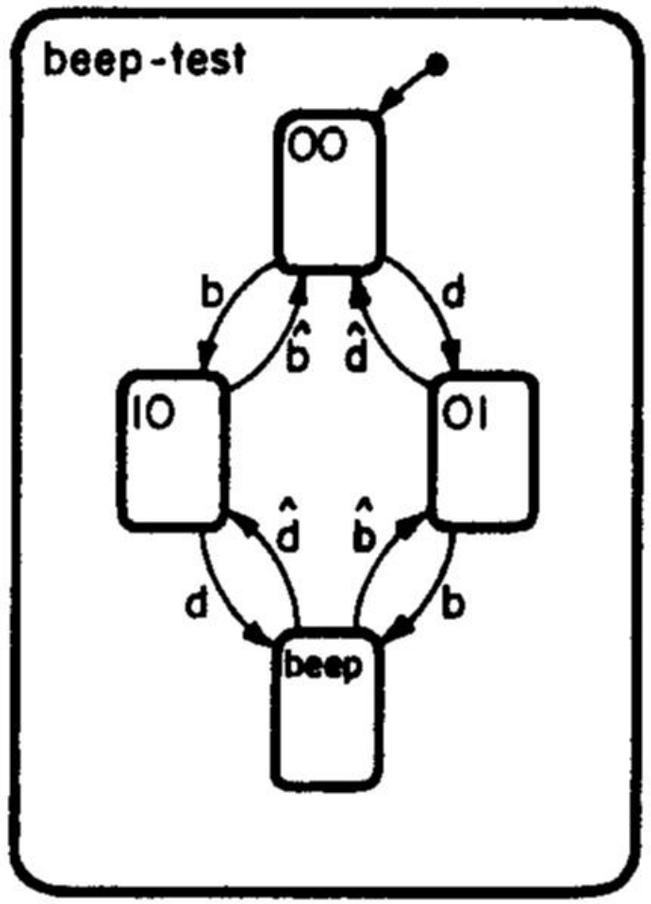

Fig. 29.

Pattern for solving race condition

"b" and "d" pressed "simultaneously".
Which is seen first?
This pattern sorts the problem out.

Skip

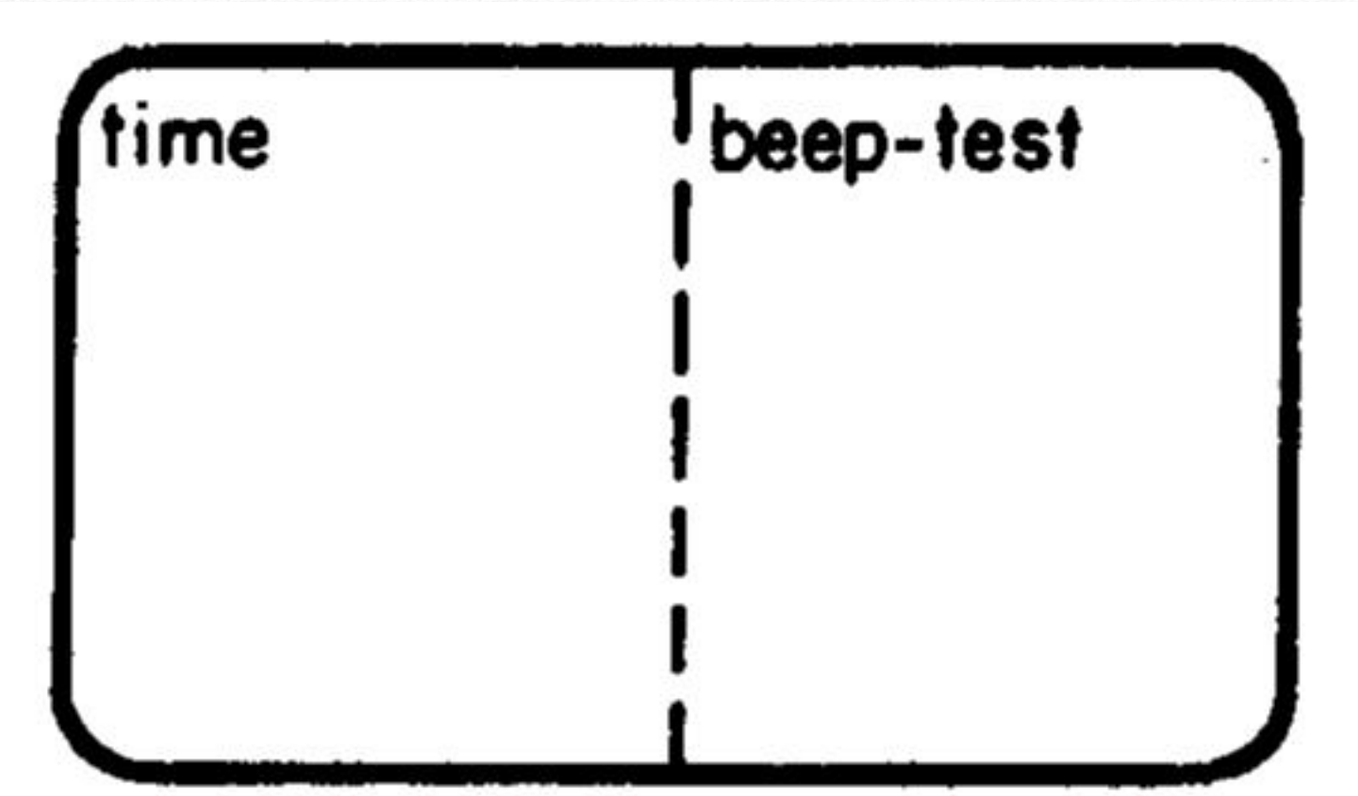

Fig. 30.

Author's idea of a reasonable design for 'time' vs. 'beep-test'. See Fig. 31 for actual.

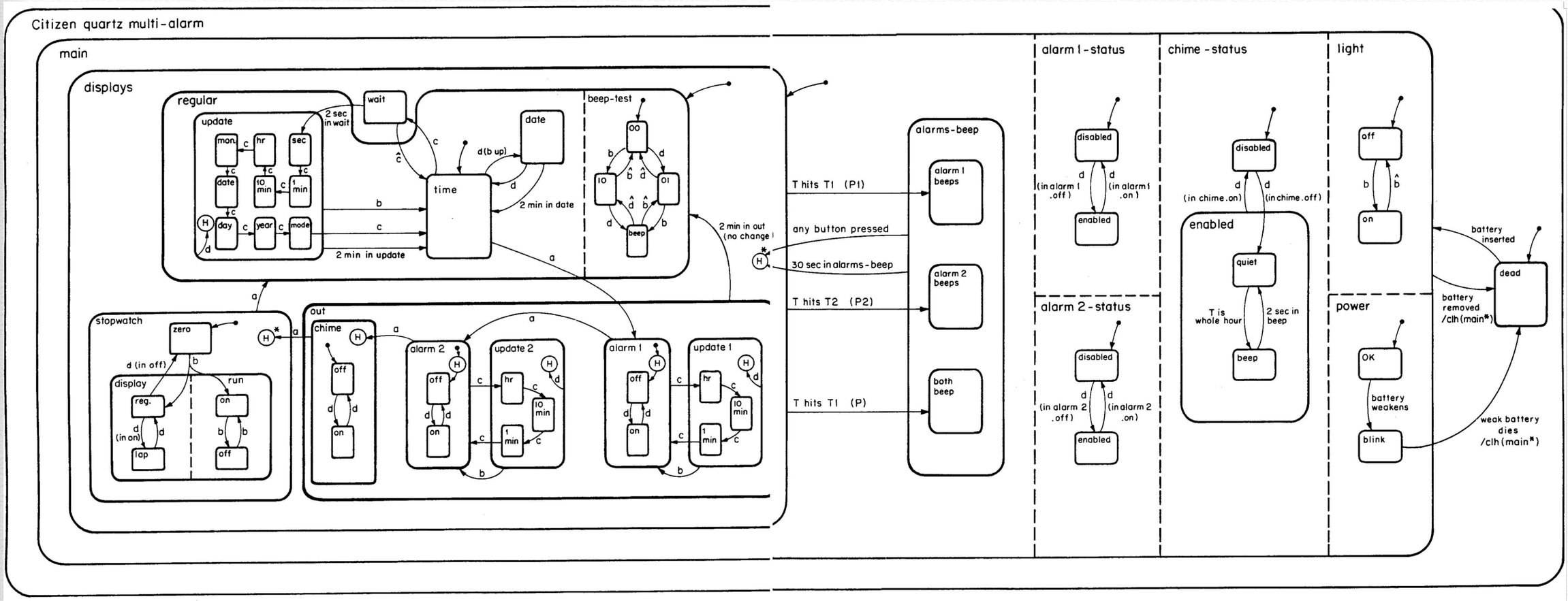

Fig. 31

Full Diagram for Digital Watch

N.B. ‘beep-test’ is valid in ‘date/time/update’,
but not in ‘wait’ - hence, notch in ‘regular’

N.B. Citizen Documentation claims that
‘beep-test’ and ‘light’ work the same,
yet author found differences.

Skip

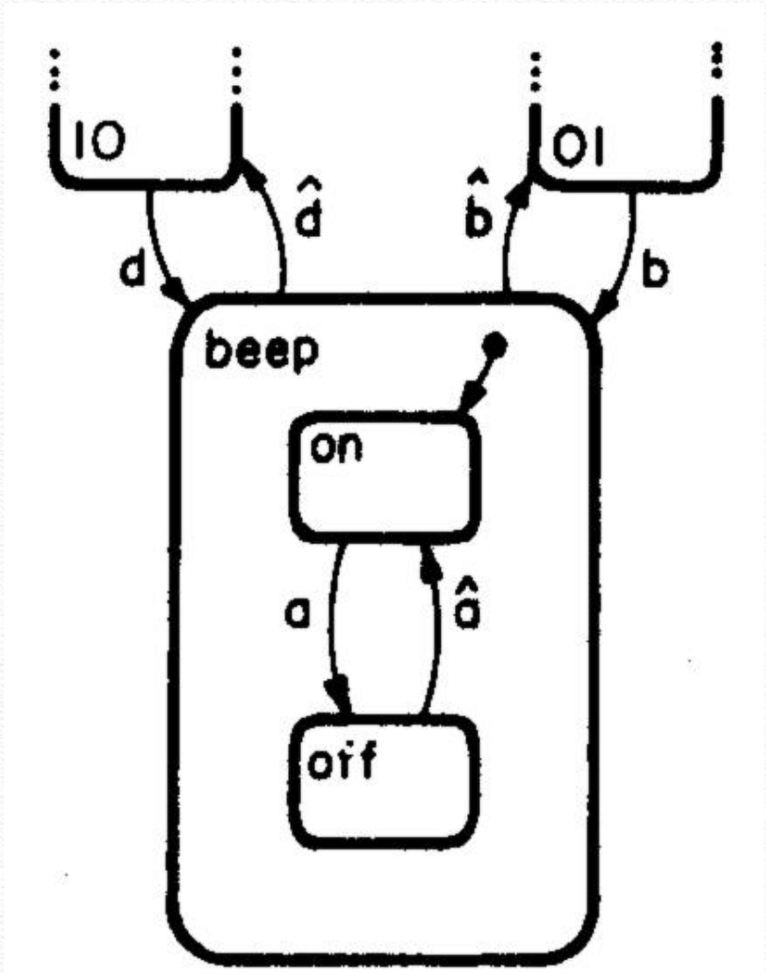

Fig. 32.

<u>Anomaly</u>

Pressing “a-down” during ‘beep’ stops beeping until “a-up”

4. Additional Statechart Features

Features that were not shown in Watch example

- Conditional
- Selection
- Timeout
- Unclustering

Skip

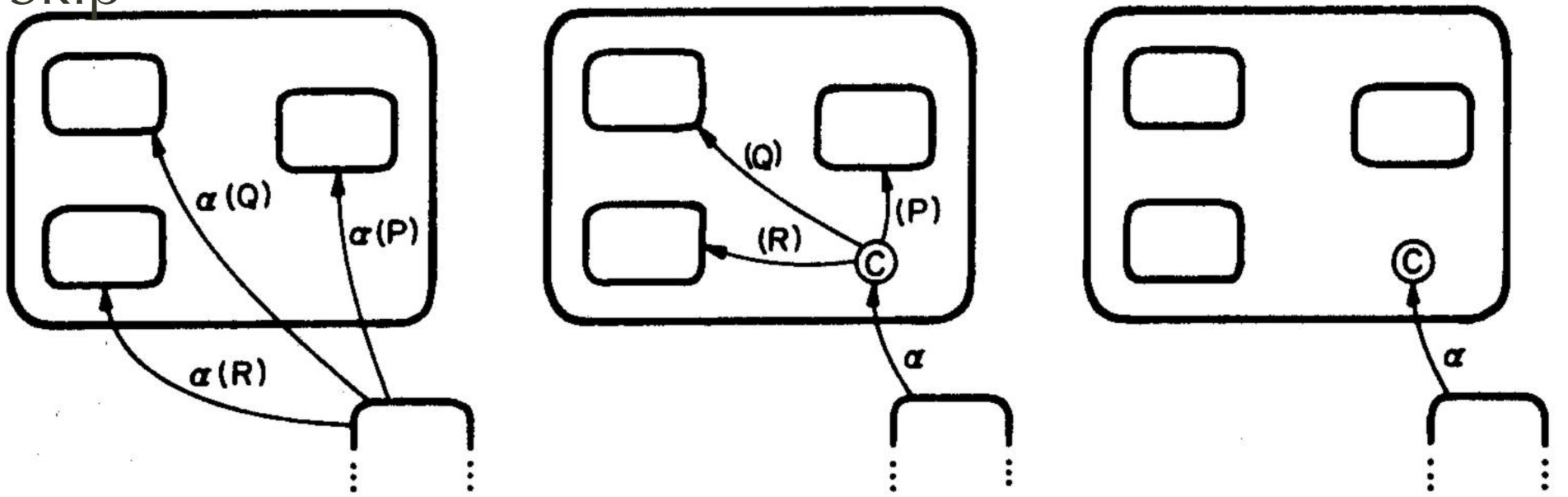

Fig. 33.

Condition & Selection

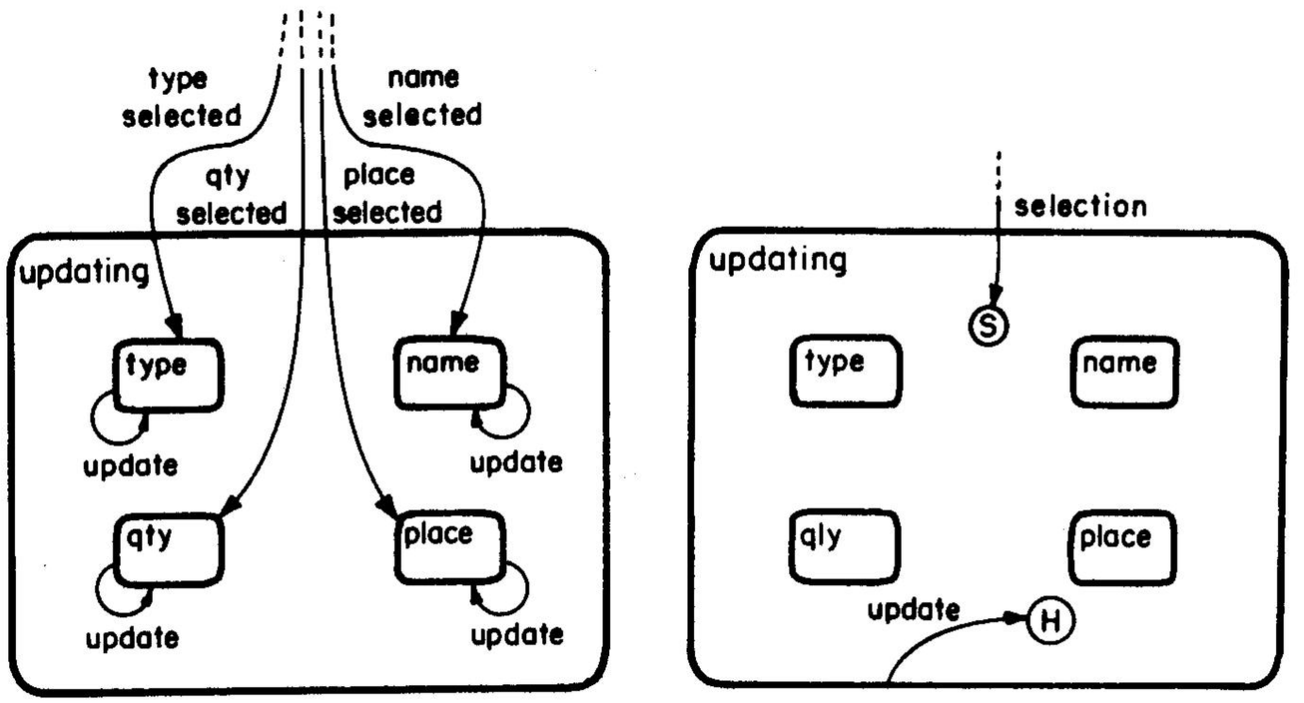

Fig. 34.

Skip

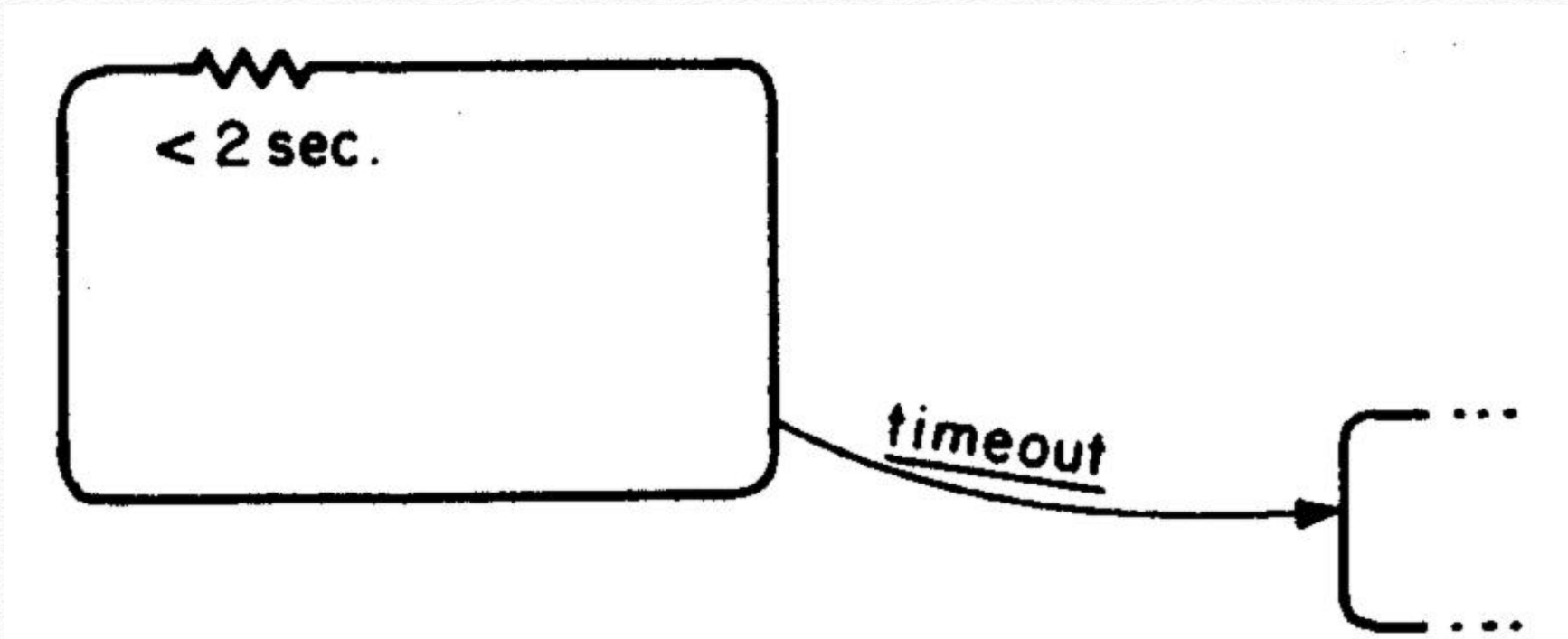

Fig. 35.

Timeout state

(lower and upper bound)

Skip

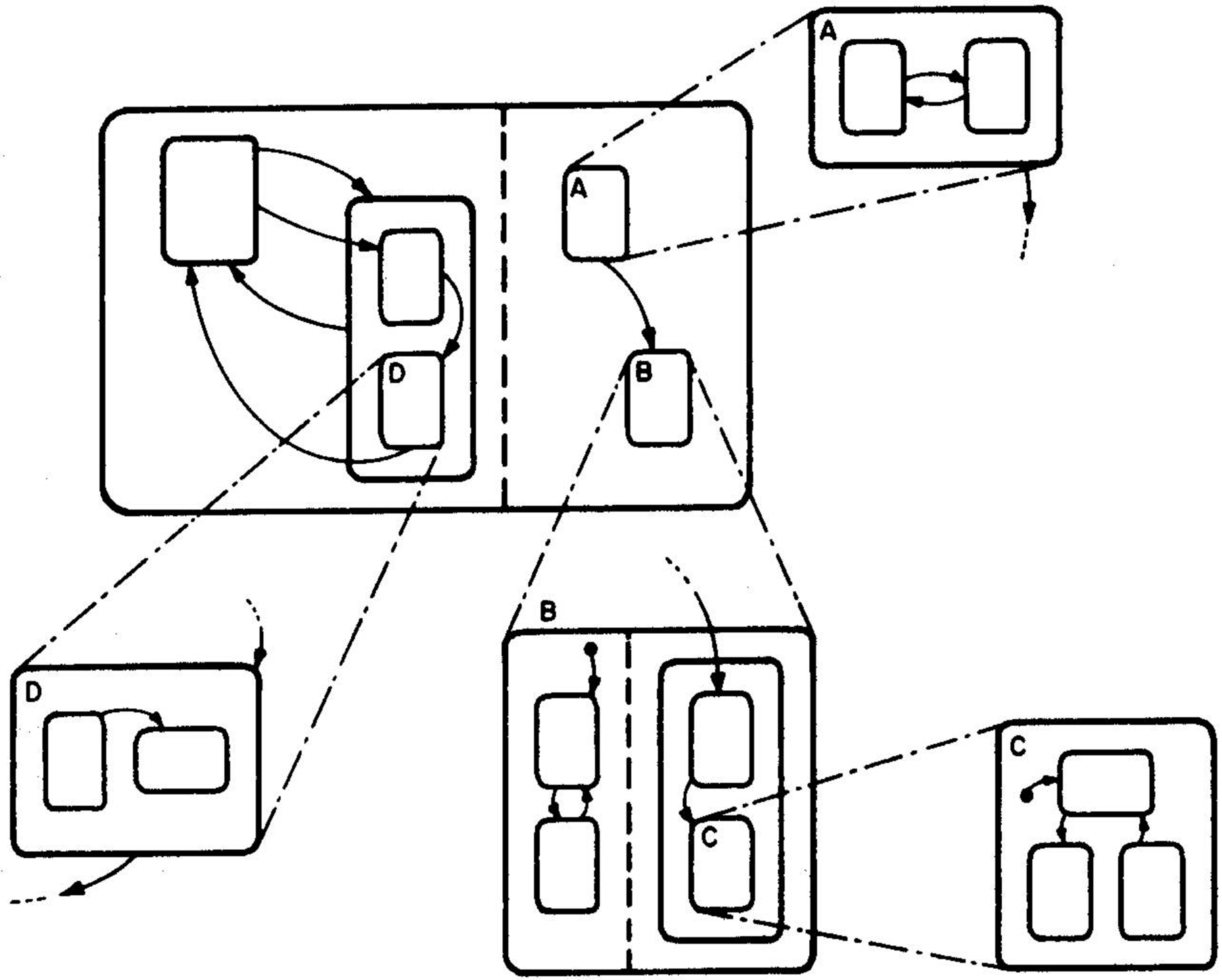

Fig. 36.

Unclustering

5. Actions and activities

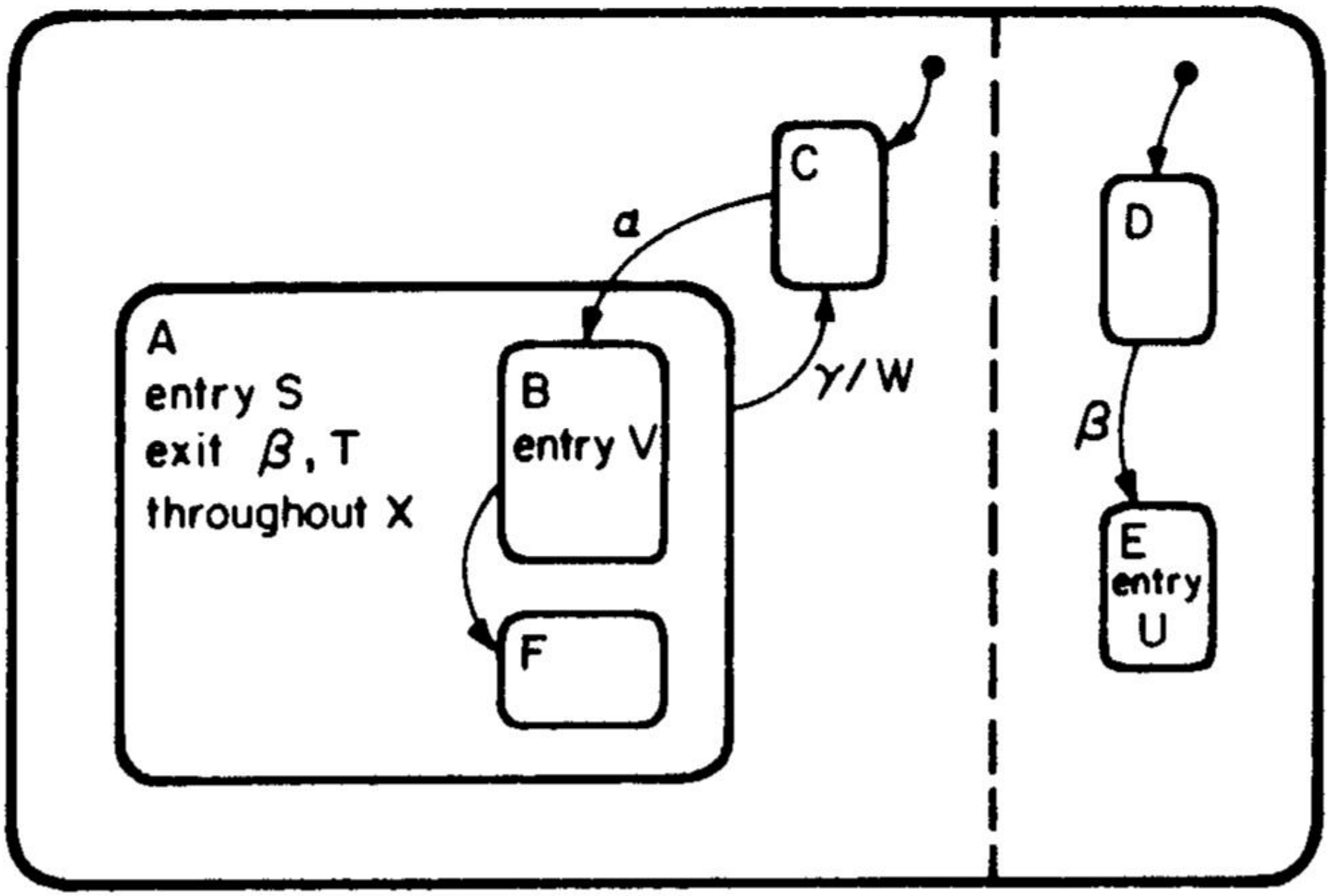

Fig. 37.

Entry & Exit Code

In state C, event *alpha* will cause execution of
"entry S", "throughout X" and "entry V"
And B->F will not cause S to be eval'ed again

6. Possible Extensions to the formalism

- Parameterized states
- Overlapping states
- D.R.Y.
- Incorporating temporal logic
- Recursive states
- Probabilistic states

Skip

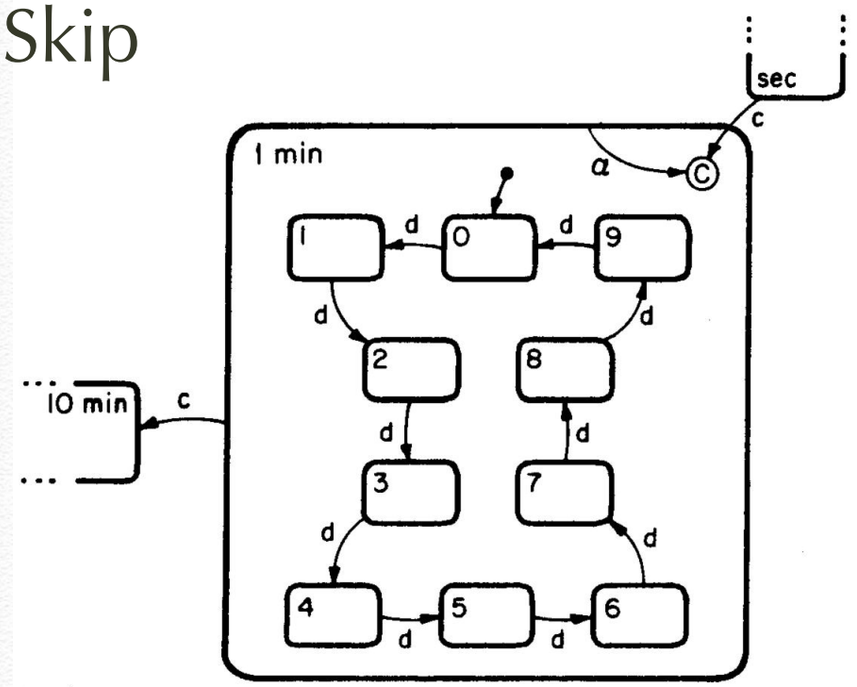

Fig. 38.

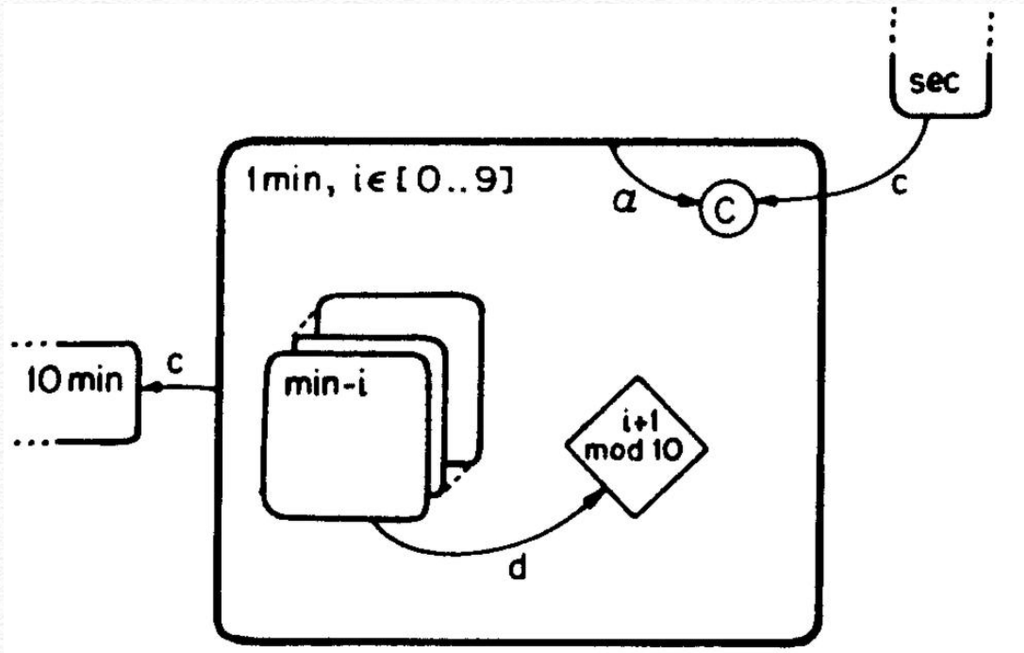

Fig. 39.

Parameterized States

Fig. 39 parameterizes Fig. 38
Fig. 40 parameterizes 1000 telephones

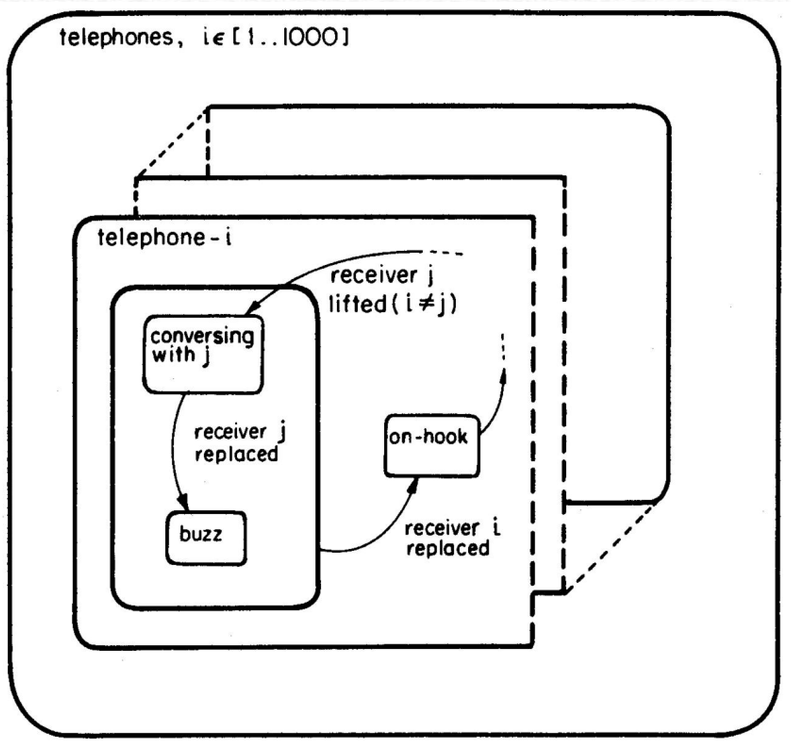

Fig. 40.

Skip

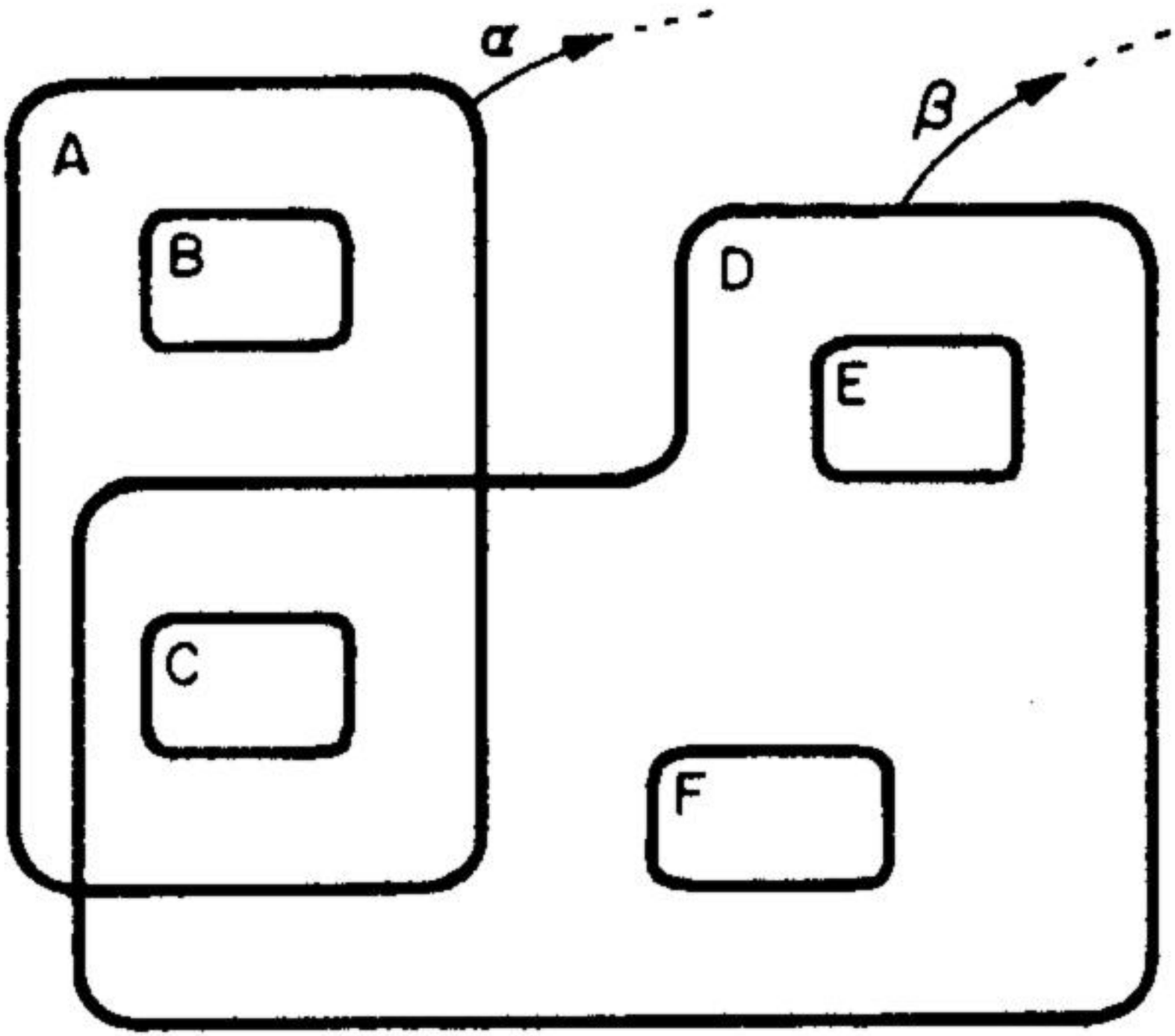

Fig. 41.

Skip

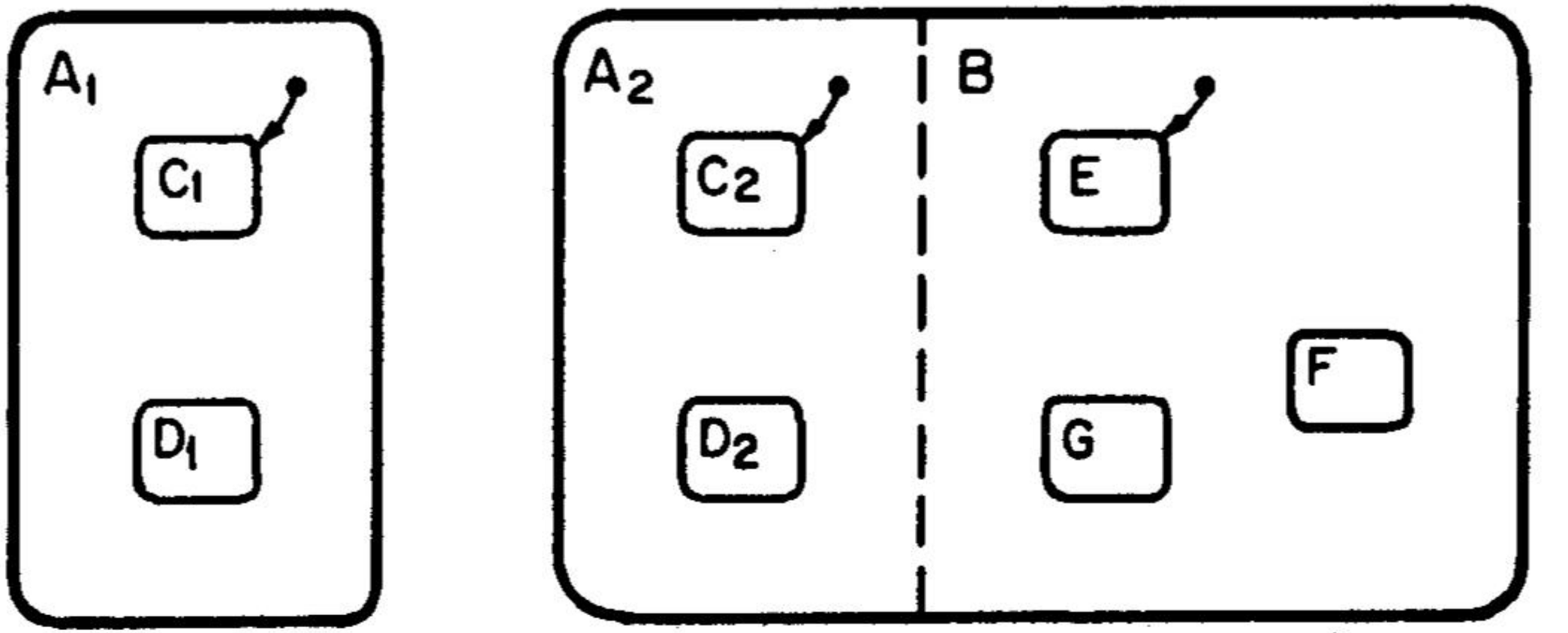

Fig. 42.

A1 and A2 are the same.

Skip

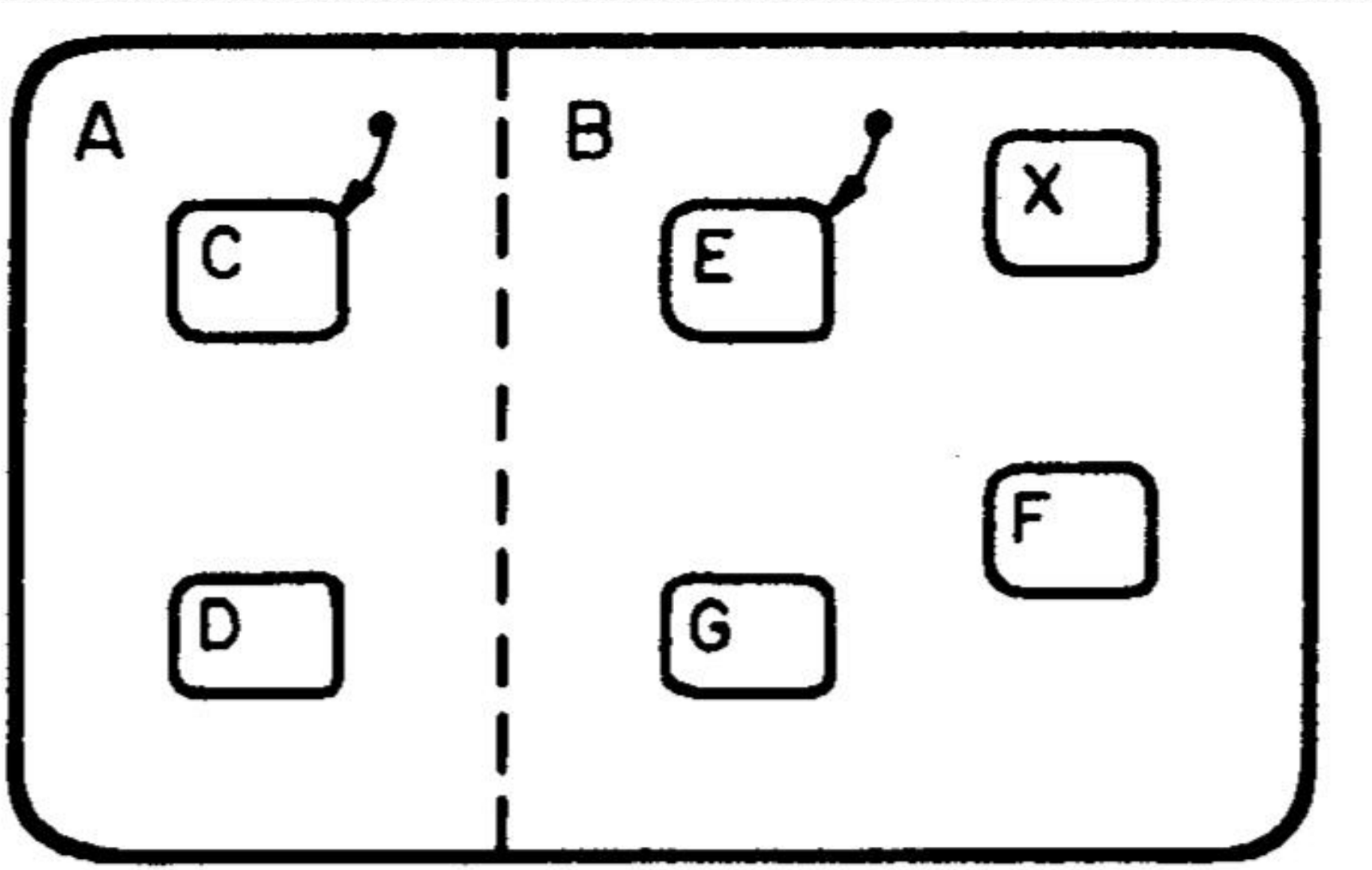

Fig. 43.

Skip

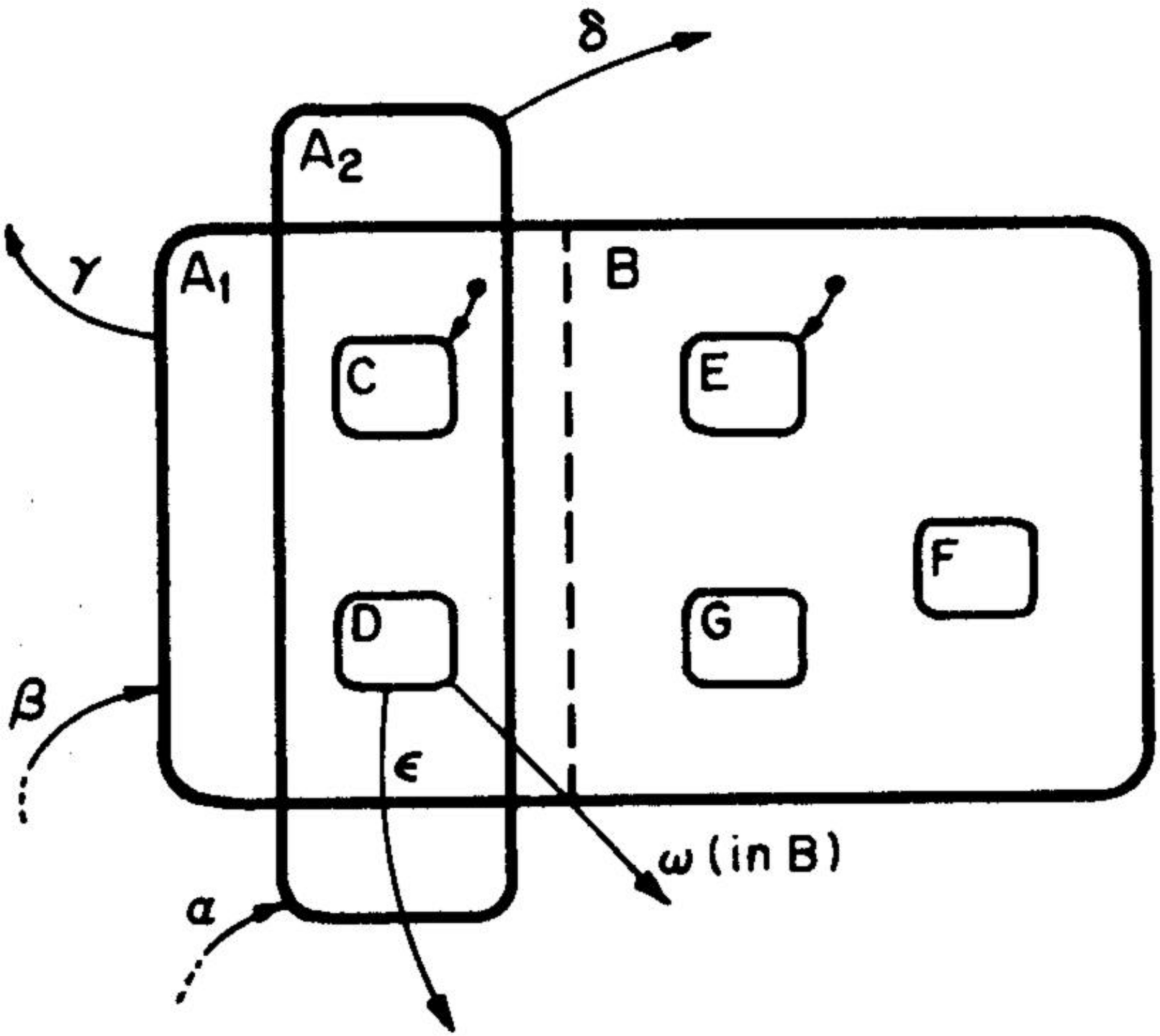

Fig. 44.

D.R.Y.

Skip

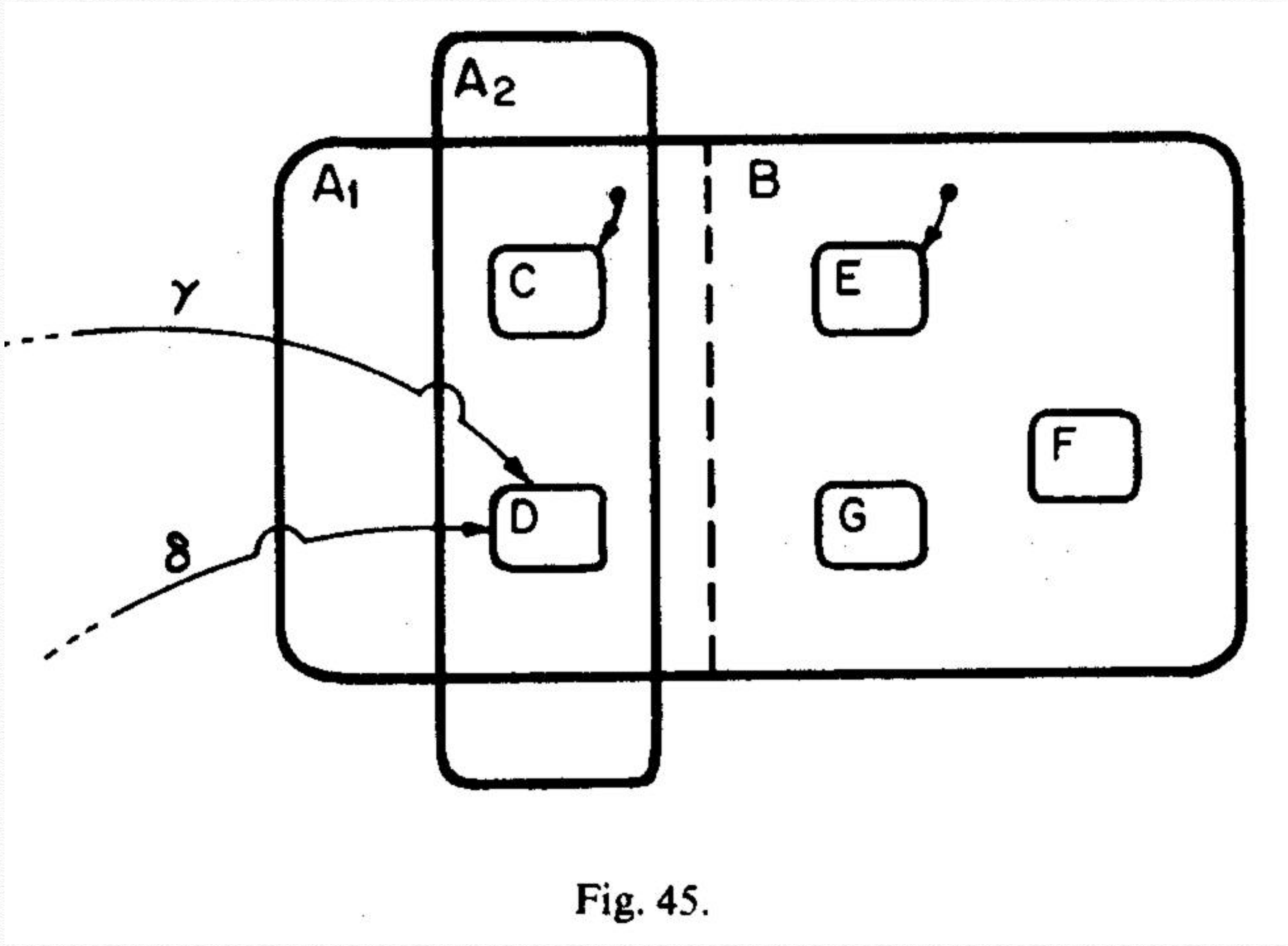

Fig. 45.

Delta transition "skips over" A1.

Skip

6.3 Incorporating temporal logic

ex. (not(in *chime)* and not(in *dead*)) since (in *chime.on*)

6.4 Recursive and probabilistic statecharts

7. Semantics of statecharts

Broadcast
Micro-steps

See [15]

Skip

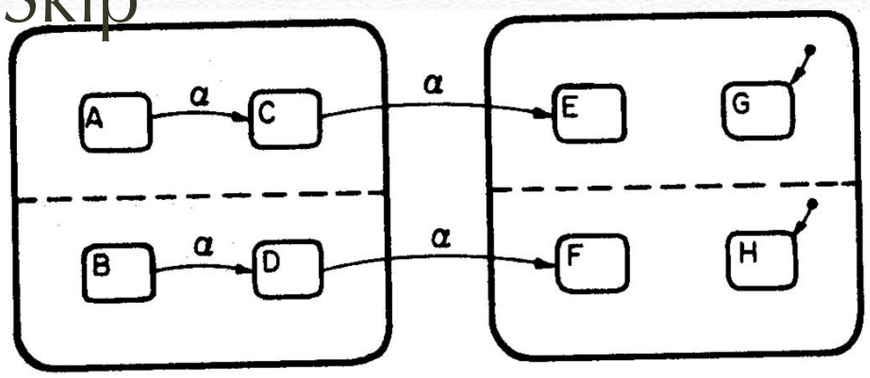

Fig. 46.

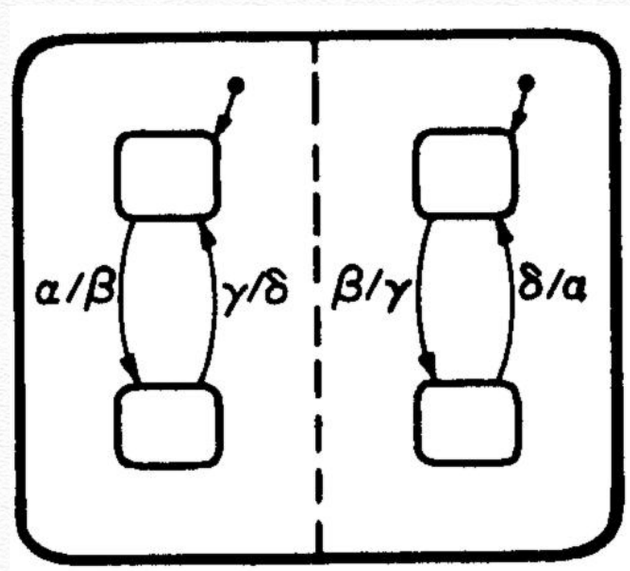

Fig. 47.

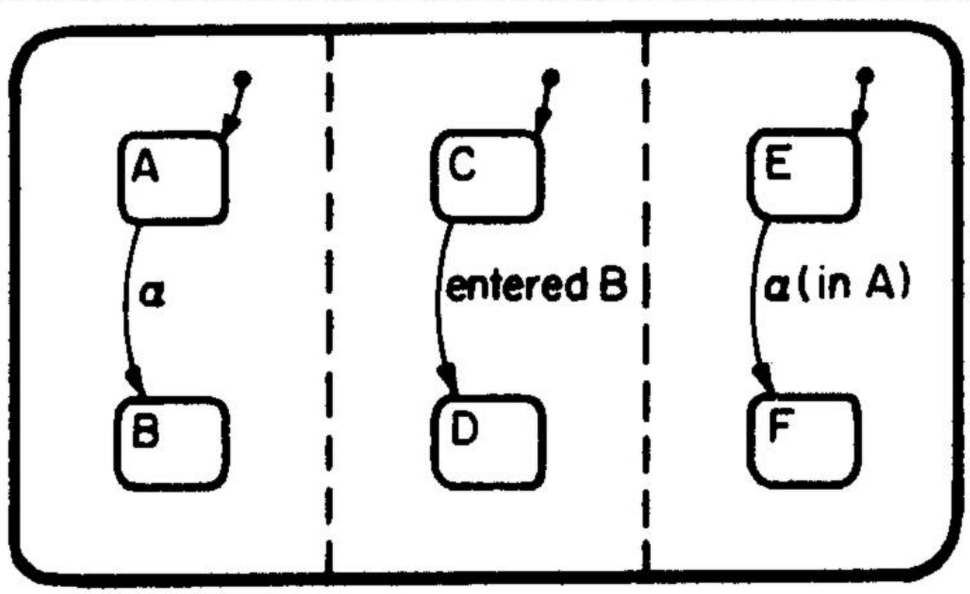

Fig. 48.

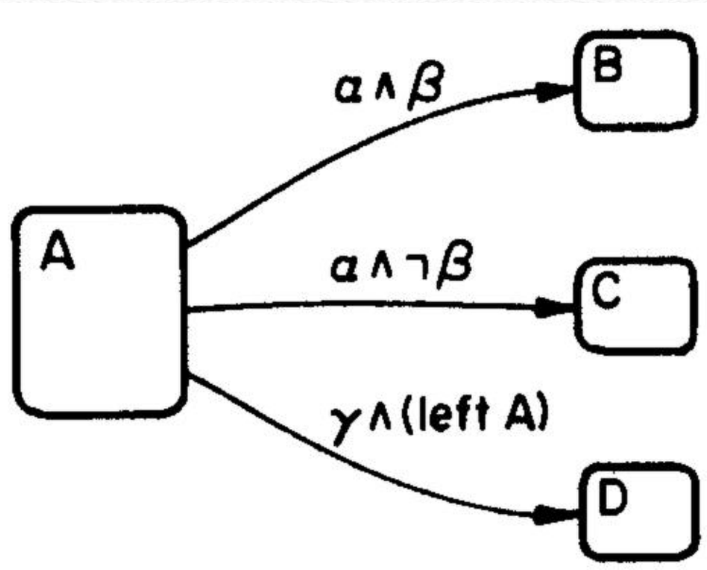

Fig. 49.

Various Semantic Issues

8. Related Work

- state explosion problem
- SDL - not hierarchical
- ATNs
- Petri nets
- CCS
- CSP
- ESTEREL
- Sequence Diagrams

9. Practical experience and implementation

(dated?)
STATEMATE1
I-Logix
IBM Rational
UML 2

My Experience

It is easy to compile diagrams. Glyphs[+] == {rect, arrow, text, dot}.
Inference (Prolog, minikanren?, pattern-matching?)
derives all other properties.

Only compiled code (diagrams) is meaningful,
Comments don't work.

Compilation. (Modeling is not compilation).

Errors are not special. Errors are events. (No need for throw/catch).

Notation[++] is understandable by "management" (kind-of Agile?)

Structured control of state.
(=> structuring other aspects, like spaghetti message-passing)

+ Code uses glyphs not pixels, e.g. a-z, A-Z, 0-9 etc.
++ See also DRAKON

My Experience (con't)

Concurrency can be lifted to another notation.

Other resources (recently discovered):

https:/statecharts.github.io

w3.org/TR/scxml/

(dragon-editor.sourceforge.net)

paultarvydas@gmail.com

https://github.com/guitarvydas