طراحی سیستم‌های تعبیه‌شده
Embedded System Design

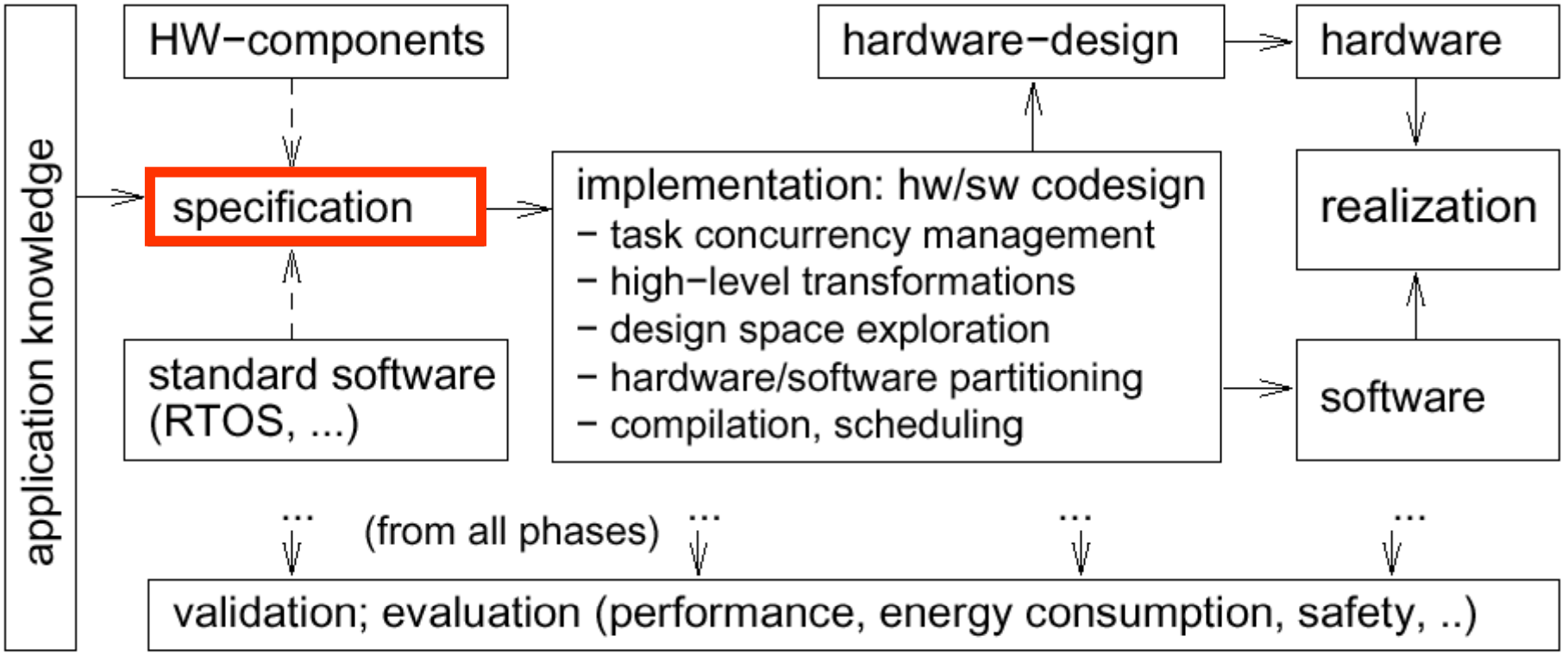
فصل دوم ـ قسمت اول

مشخص‌سازی
Specifications

کاظم فولادی
دانشکده‌ی مهندسی برق و کامپیوتر
دانشگاه تهران

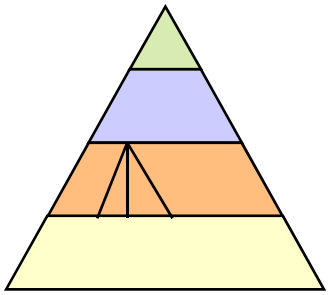
kazim@fouladi.ir

Specifications

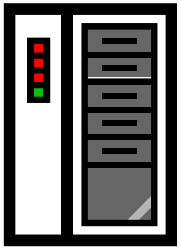


مشخص‌سازی

نیازمندی‌های لازم برای روش مشخص‌سازی



proc
proc
proc



- **سلسله‌مراتب Hierarchy**

انسان نمی‌تواند سیستم‌های حاوی بیش از چند شیء را درک کند.

اکثر سیستم‌های واقعی تعداد بیشتری شیء دارد.

راه‌حل: استفاده از **سلسله‌مراتب**

سلسله‌مراتب رفتاری behavioral

حاوی اشیای لازم برای توصیف رفتار سیستم:

مانند: حالت‌ها، رویدادها، سیگنال‌های خروجی

سلسله‌مراتب ساختاری structural

توصیف چگونگی تشکیل سیستم از اجزای فیزیکی

مانند: سیستم تعبیه‌شده متشکل است از پردازنده، حافظه، حسگر، ...

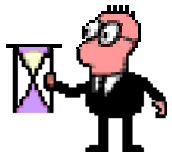
پردازنده متشکل است از ثبات، مالتی‌پلکسر، جمع‌کننده، ...

جمع‌کننده متشکل از گیت، ...

مشخص‌سازی

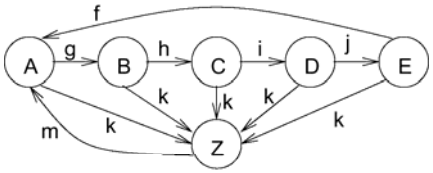
نیازمندی‌های لازم برای روش مشخص‌سازی (ادامه)

- **رفتار زمانی (Timing bahavior)**



- **رفتار حالت ـ گرا (State-oriented behavior)**

  لازم برای توصیف سیستم‌های واکنشی آتاماتای کلاسیک ناکافی است.

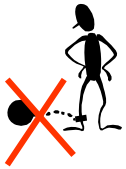


- **اداره کردن رویدادها (Event-handling)**

  رویدادهای داخلی: ناشی از اجزای سیستم

  رویدادهای خارجی: ناشی از محیط

- **عدم وجود مانع برای پیاده‌سازی کارامد**



مشخص‌سازی

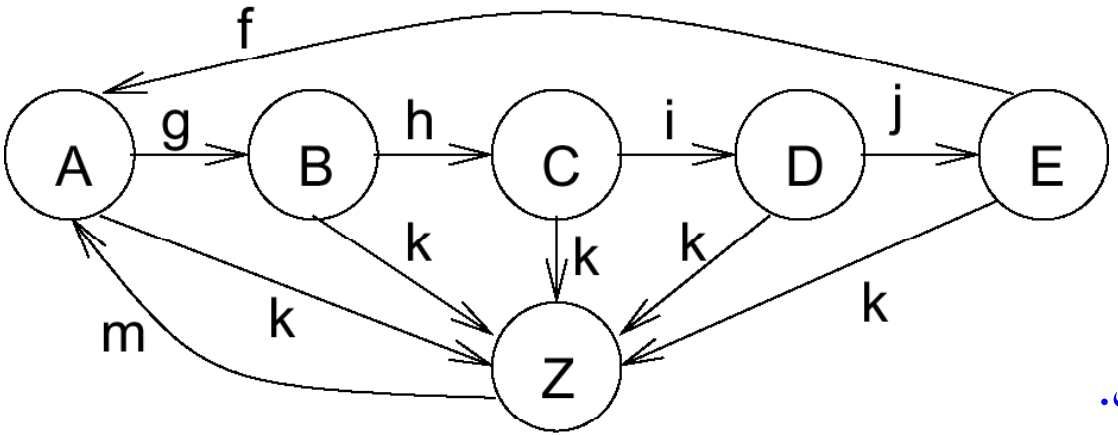
نیازمندی‌های لازم برای روش مشخص‌سازی (ادامه)

- **پشتیبانی از طراحی سیستم‌های قابل اتکا**

  وجود معنی غیر مبهم، ...

- **رفتار استثنا ـ گرا (Exception-oriented behavior)**

  لازم نباشد که مجبور باشیم استثناها را برای هر حالت تعریف کنیم.



برچسب k بیانگر استثنا است.

مشخص‌سازی

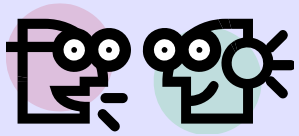
نیازمندی‌های لازم برای روش مشخص‌سازی (ادامه)

- **همروندی (Concurrency)**

  سیستم‌های دنیای واقعی همروند هستند

- **همگام‌سازی و ارتباطات (Synchronization & communication)**

  مؤلفه‌های سیستم بایستی با هم ارتباط برقرار کنند.



- **حضور عناصر برنامه‌سازی**

  مانند اعمال محاسباتی، حلقه‌ها، فراخوانی توابع، . . .

- **قابلیت اجرا (Executability)**
- **پشتیبانی از طراحی سیستم‌های بزرگ** (شیءگرایی)
- **پشتیبانی زمینه‌های کاربردی خاص**

مشخص‌سازی

نیازمندی‌های لازم برای روش مشخص‌سازی (ادامه)

- **خوانایی (Readability)**

خوانابودن برنامه برای انسان و
قابل درک بودن آن توسط ماشین.

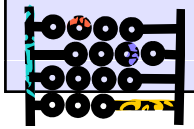
- **قابلیت حمل و انعطاف‌پذیری (Portability & flexibility)**

- **خاتمه (Termination)**

باید بوضوح مشخص باشد که در چه زمانی هر محاسبه کامل می‌شود.

- **پشتیبانی از دستگاه‌های I/O غیراستاندارد**
- **ارائه‌ی ویژگی‌های غیرکارکردی**

تحمل‌پذیری در برابر نقص، مصرفی بودن، وزن، اندازه، کاربرپسندی، سازگاری الکترومغناطیسی EMC، قابلیت گسترش، توان مصرفی، عمر



- **مدل محاسباتی مناسب**

مشخص‌سازی

مدل‌های محاسباتی ـ تعریف

Computational Models

مدل‌های محاسباتی موارد زیر را تعریف می‌کنند:

- **مؤلفه‌ها**

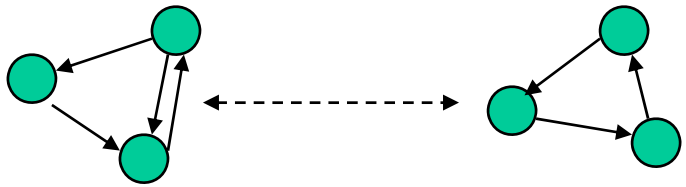
  زیرروال‌ها، پردازش‌ها، نخ، . . .
- **مکانیزم تعامل مؤلفه‌ها (قراردادهای ارتباطی)**
- تبادل پیام؟ قرار ملاقات؟ . . .
- **و احتمالاً آنچه مؤلفه‌ها در مورد یکدیگر می‌دانند**

  مانند متغیرهای سراسری: رفتار ضمنی مؤلفه‌های دیگر، . . .

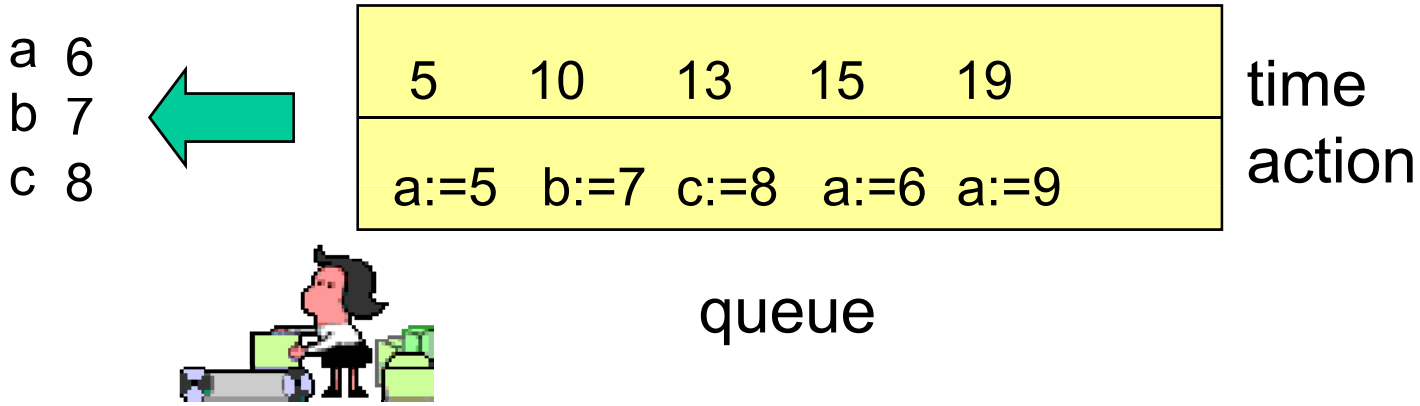
مشخص‌سازی

مدل‌های محاسباتی ـ نمونه‌ها

- **ماشین‌های متناهی‌حالت ارتباطی**:(CFSMs)



- **مدل گسسته رویداد** (Discrete event model)



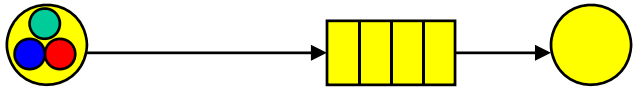
مشخص‌سازی

مدل‌های محاسباتی ـ نمونه‌ها (ادامه)

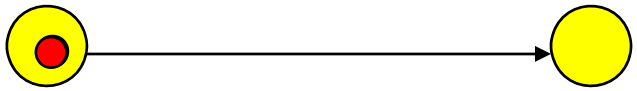
- معادلات دیفرانسیل (Differential equations)

$$\frac{\partial^2 x}{\partial t^2} = a$$

- انتقال پیام ناهمگام (Asynchronous message passing)



- انتقال پیام همگام (Synchronous message passing)



☞ Ptolemy simulations?

مشخص‌سازی

مدل‌های محاسباتی

واقعیت:

هیچ زبانی همه‌ی نیازمندی‌های فوق را رعایت نمی‌کند

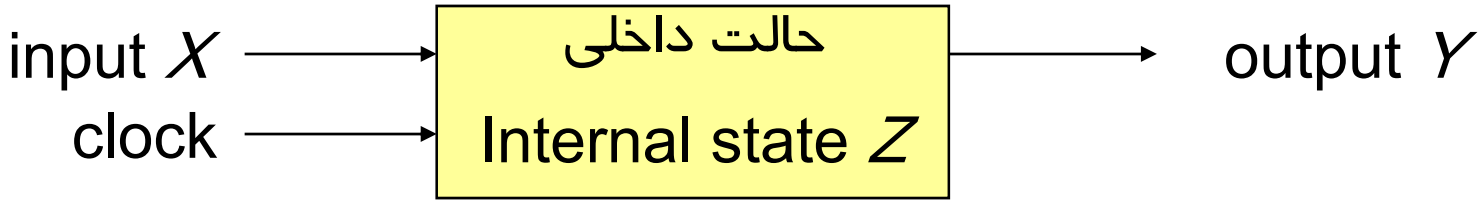
⇐ نیاز به مصالحه

StateCharts

جانشین آتاماتای کلاسیک

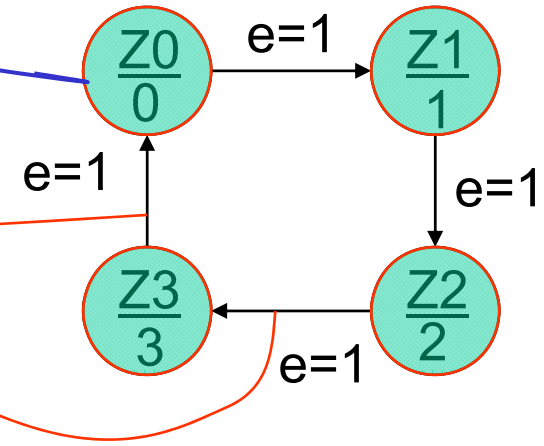
automata = finite state machines (FSMs)

آتاماتای کلاسیک:



Next state Z^+ computed by function δ
Output computed by function λ

- **Moore**-automata:
  $Y = \lambda(Z)$; $Z^+ = \delta(X, Z)$
- **Mealy**-automata
  $Y = \lambda(X, Z)$; $Z^+ = \delta(X, Z)$



StateCharts
جانشین آتاماتای کلاسیک

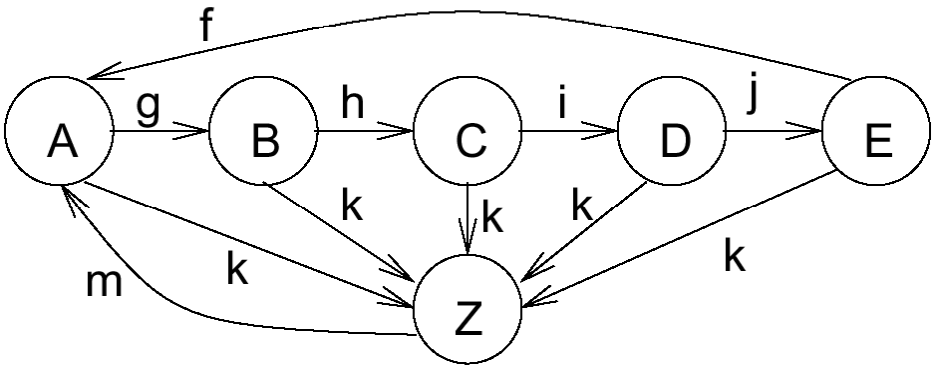
آتاماتای کلاسیک برای توصیف سیستم‌های پیچیده مفید نیست:

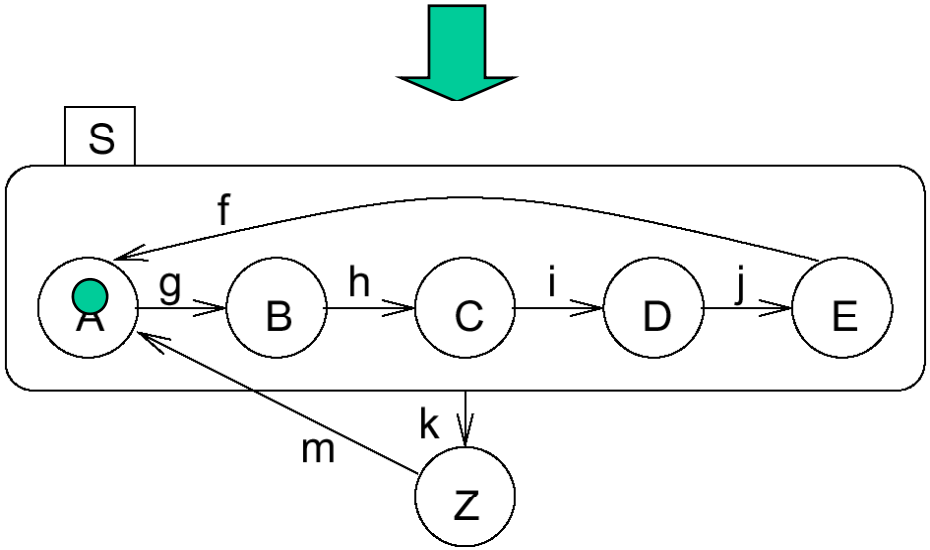
(گراف‌های پیچیده به راحتی توسط انسان درک نمی‌شوند)

وارد کردن سلسله‌مراتب به آتاماتای کلاسیک ⇐ StateCharts

StateCharts

وارد کردن سلسله‌مراتب



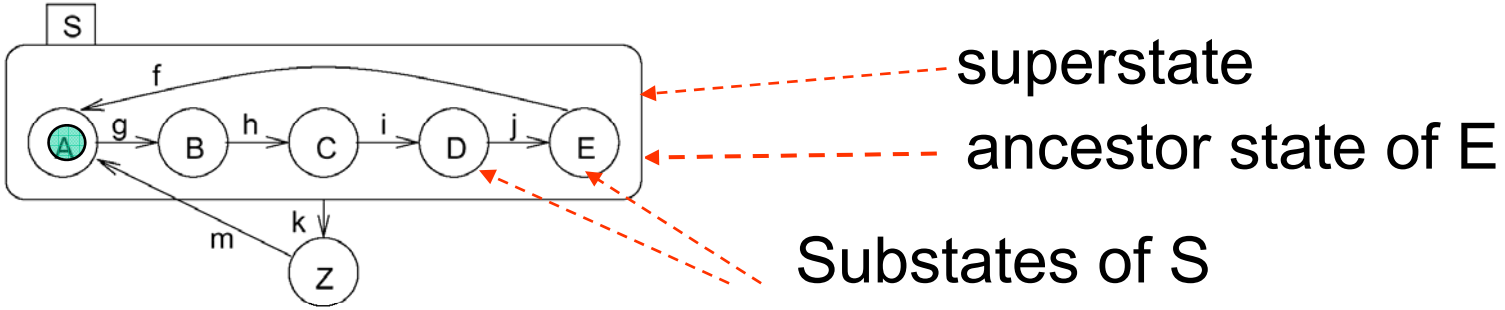


اگر S **فعال** (**active**) باشد،
FSM دقیقاً در یکی از
زیرحالتهای S قرار دارد:
(A یا B یا C یا . . .)

StateCharts

چند تعریف

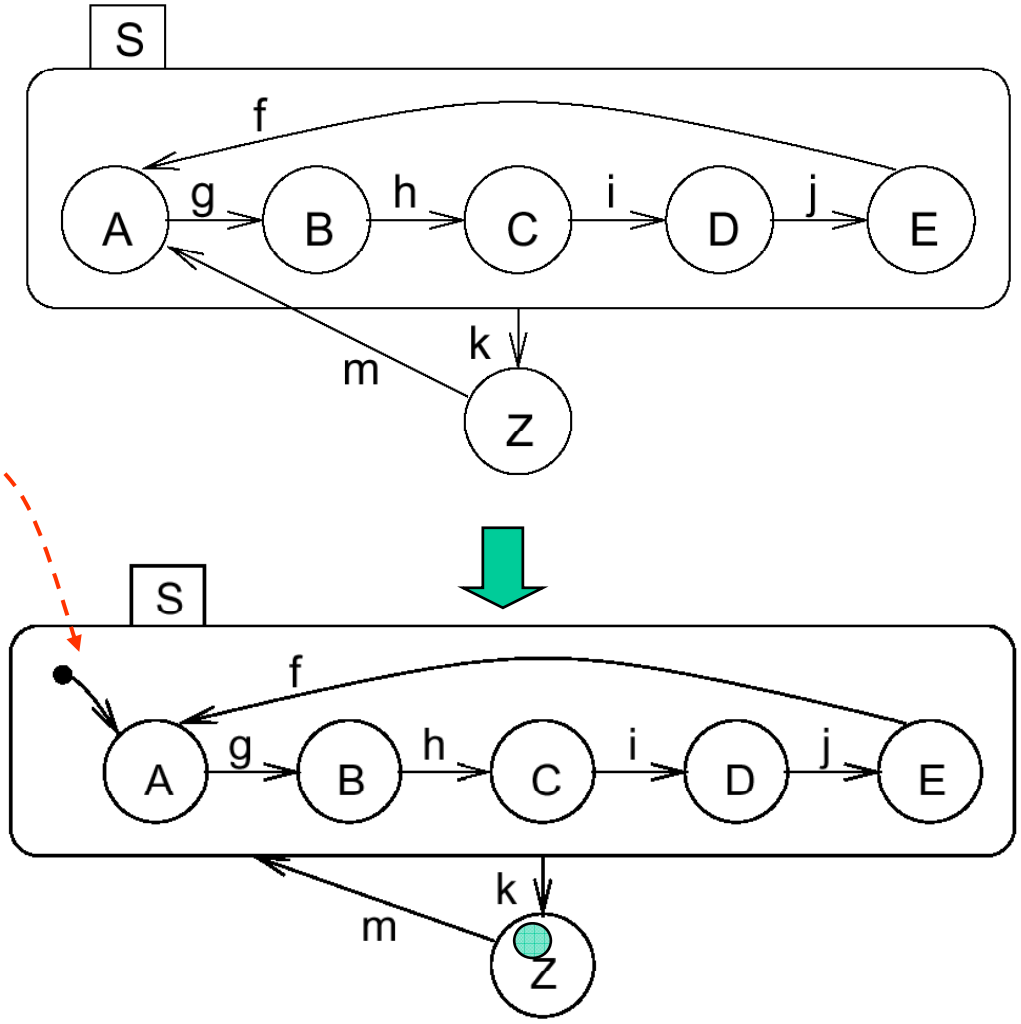
- حالت‌های فعلی FSMها حالت‌های **فعال (active)** هم نامیده می‌شوند.
- حالت‌هایی که از حالت‌های دیگر تشکیل نشده‌اند، **حالت پایه** نام دارند.
- حالت‌های حاوی حالت‌های دیگر **ابرحالت (super-state)** خوانده می‌شوند.
- ابرحالت‌های حاوی حالت پایه‌ی S، **حالات جد (ancestor)** نام دارند.
- ابرحالت S **ابرحالت OR** نام دارد، اگر در هنگام فعال بودن S تنها یکی از زیرحالت‌های آن فعال باشد.



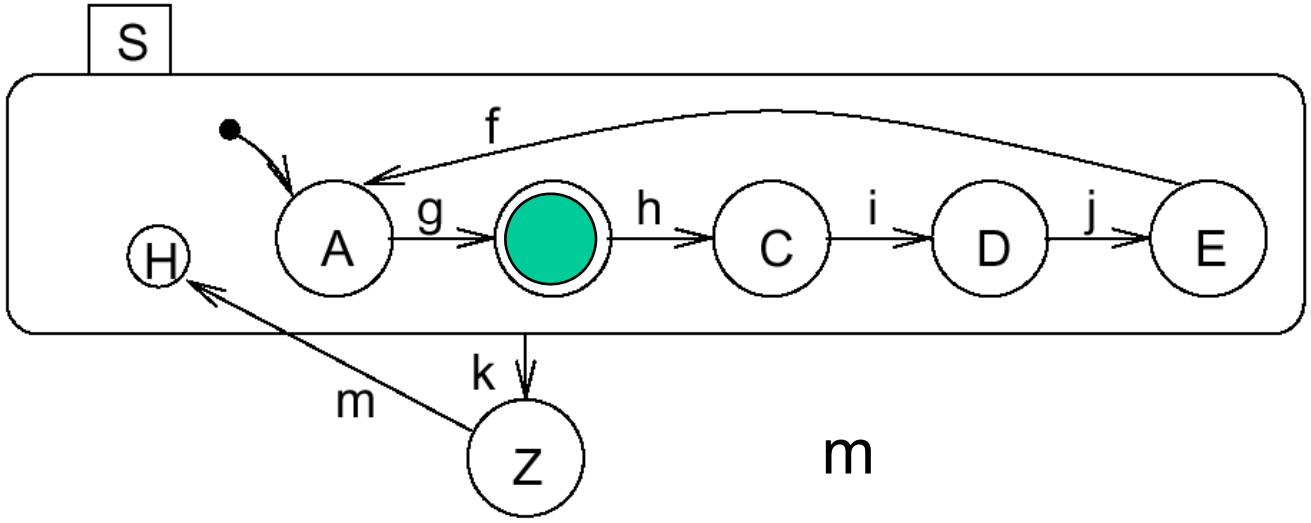
StateCharts

مکانیزم حالت پیش‌فرض

پنهان کردن ساختار
از دید دنیای خارجی ⇐
حالت پیش‌فرض (Default state)
دایره‌ی توپر زیرحالتی را نشان می‌دهد که در هنگام ورود به ابرحالت وارد آن می‌شویم.



StateCharts
مکانیزم تاریخچه



m

رفتاری متفاوت با اسلاید قبل:

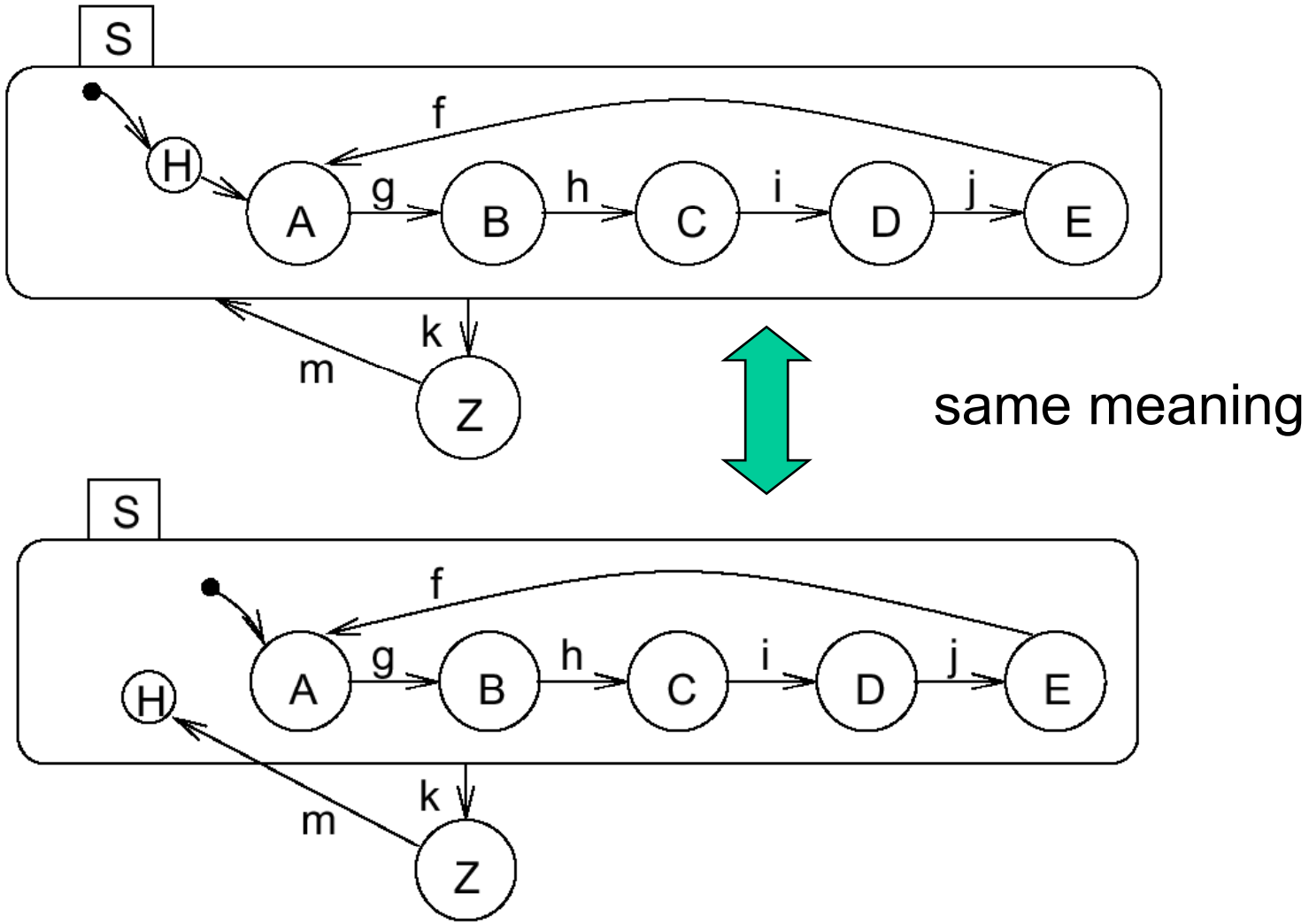
برای ورودی m، S وارد حالتی حالتی می‌شود که قبلاً در آن بوده است. (می‌تواند حالت A، B، C، D یا E باشد).

اگر برای اولین مرتبه وارد Sشویم، مکانیزم حالت پیش‌فرض استفاده می‌شود.

مکانیزم‌های تاریخچه و حالت پیش‌فرض می‌توانند به صورت سلسله‌مراتبی به کار روند.

StateCharts

ترکیب مکانیزم تاریخچه و حالت پیش‌فرض



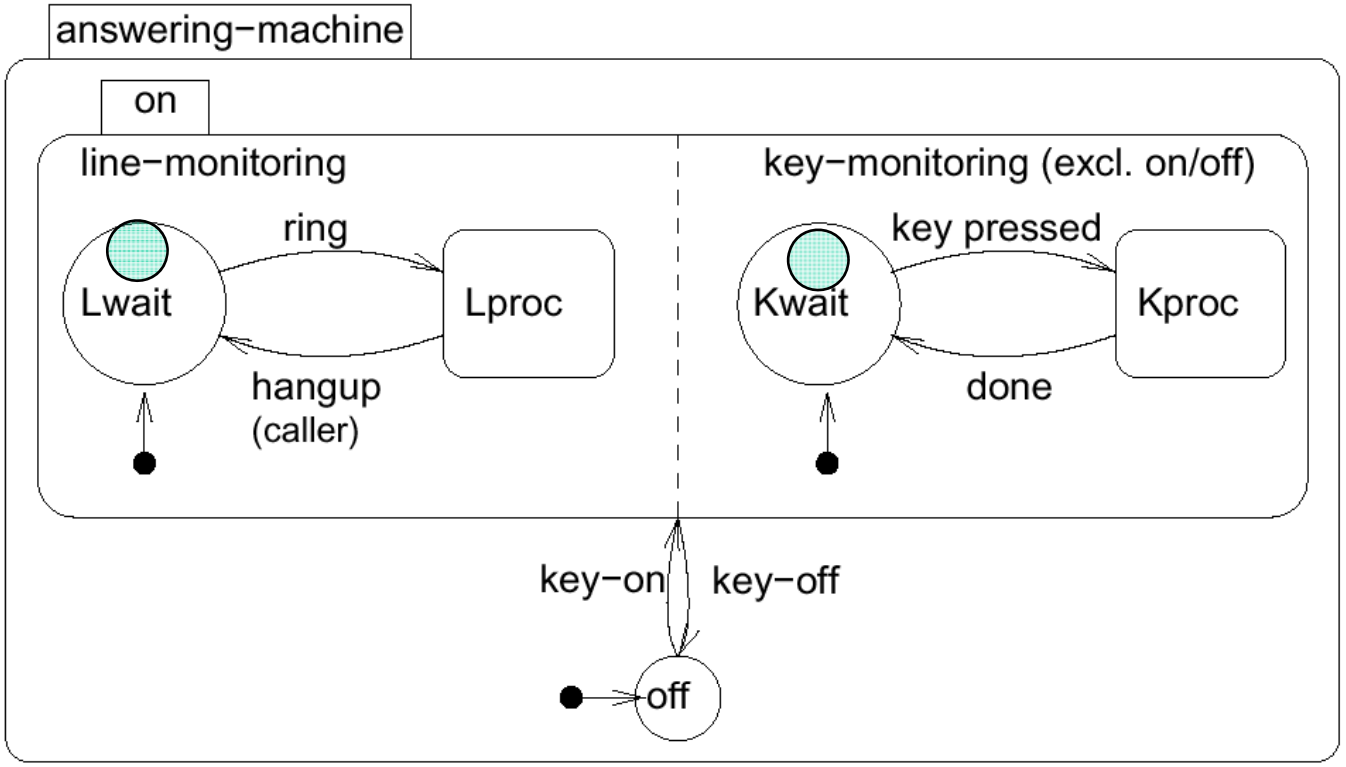
StateCharts

همروندی

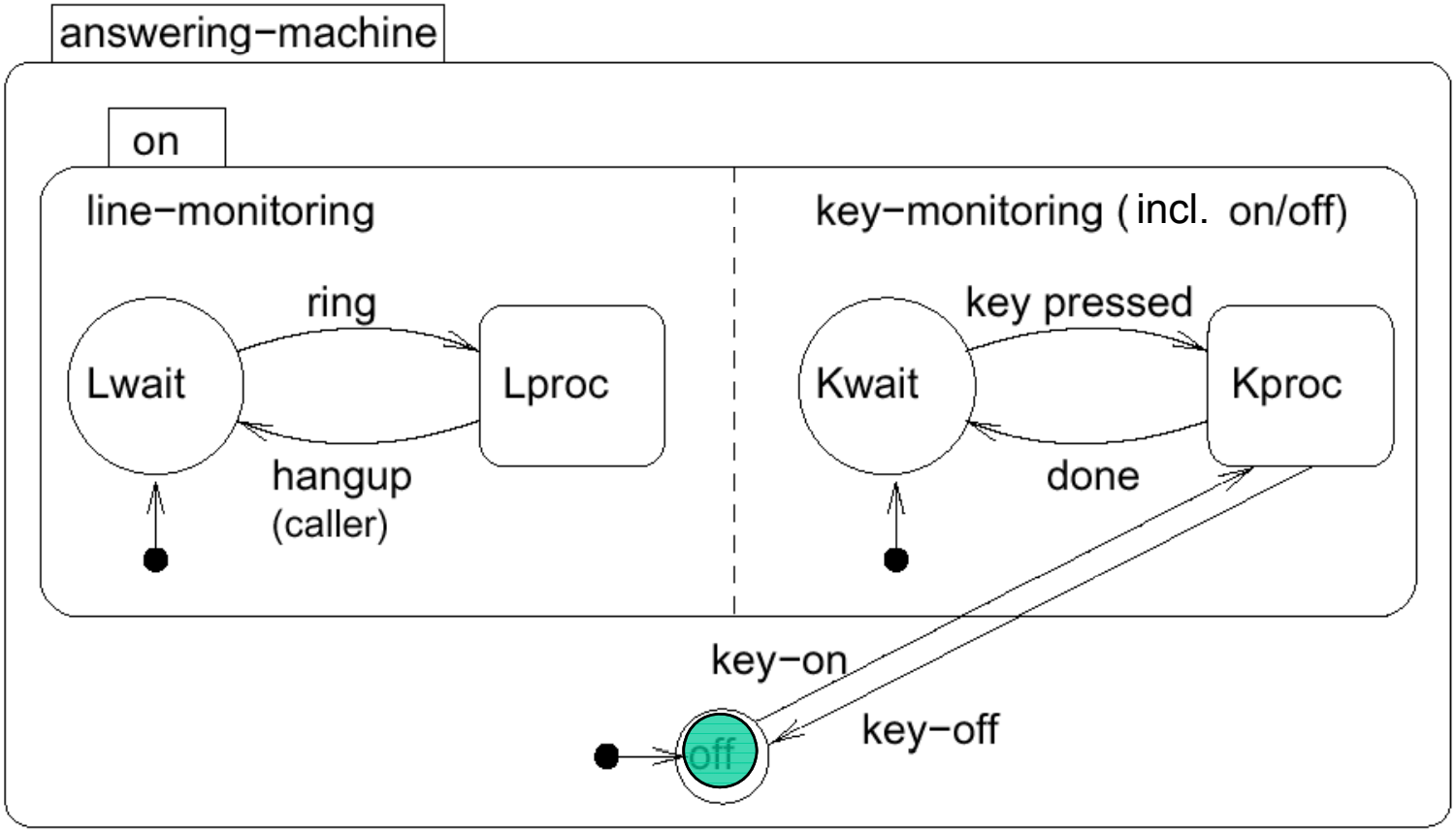
روش‌های مناسبی برای توصیف همروندی (concurrency) لازم است.

ابرحالت‌های AND (AND-super-states):

FSM در **همه‌ی** زیرحالت‌های (بی‌واسطه‌ی) آن ابرحالت قرار دارد.



StateCharts
ورود و ترک ابرحالت‌های AND



Line-monitoring and key-monitoring are entered and left, when service switch is operated.

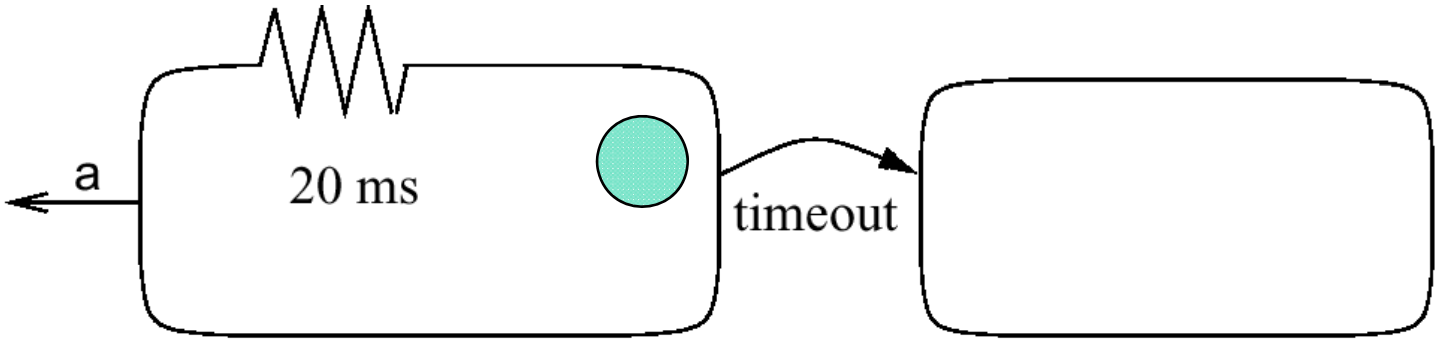
StateCharts
انواع حالت‌ها

- basic states
- AND-super-states
- OR-super-states.

StateCharts
تایمرها

تایمر: نیاز به مدل‌سازی زمان در سیستم‌های تعبیه‌شده

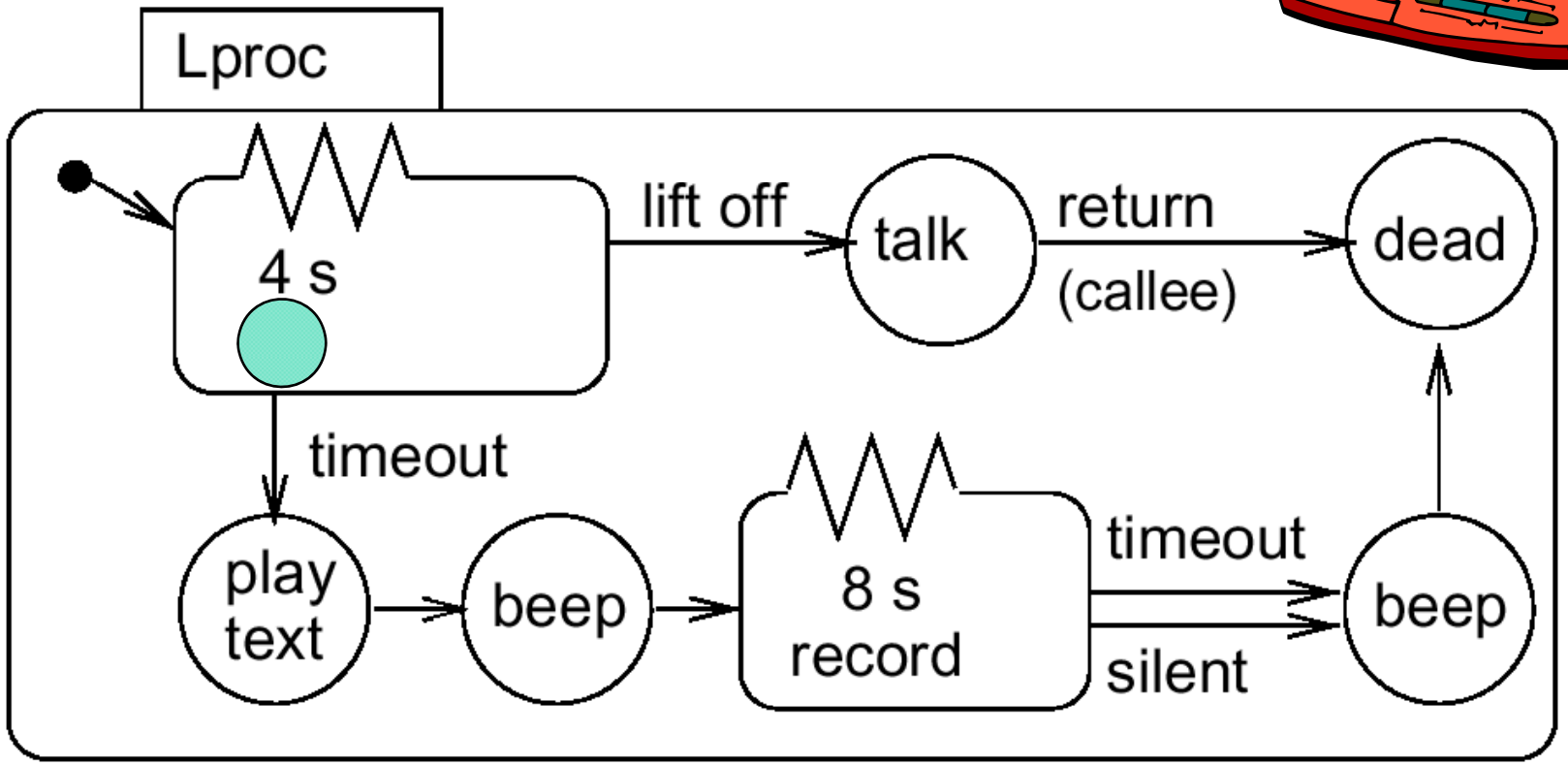
در StateCharts یال‌های خاص برای timeoutها استفاده می‌شود.



اگر رویداد a در مدت زمان 20 ms از زمان ورود به حالت سمت چپ رخ ندهد، یک timeout اتفاق می‌افتد.

StateCharts

استفاده از تایمرها در ماشین پاسخگویی



StateCharts

شکل عمومی برچسب یال‌ها

event:

- یک رویداد تنها تا ارزیابی بعدی مدل وجود دارد
- رویداد می‌تواند به صورت داخلی یا خارجی تولید شود.

condition:

- شرط براساس مقدار متغیرهایی که تا انتساب مجدد مقدار خود را حفظ می‌کنند.

reaction:

- می‌تواند یک انتساب متغیر یا ایجاد یک رویداد باشد.

مثال:

- service-off [not in Lproc] / service:=0

StateCharts

مفاهیم StateMate: مراحل شبیه‌سازی StateCharts

چگونه برچسب یال‌ها ارزیابی می‌شوند؟

طی سه مرحله:

۱. تاثیر تغییرات خارجی بر روی رویدادها و شرایط ارزیابی می‌شود.

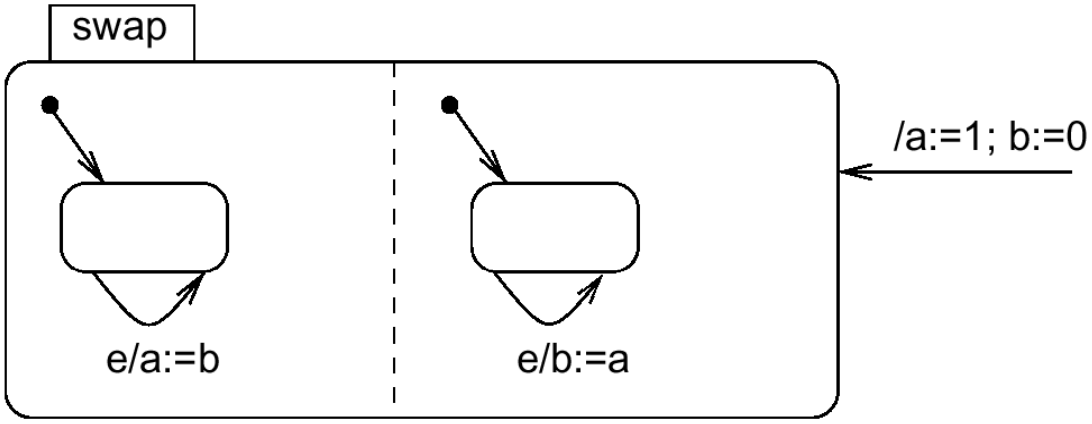
۲. مجموعه‌ی گذرهایی که باید در مرحله‌ی فعلی انجام شود و سمت راست انتساب‌ها محاسبه می‌شود.

۳. گذرها انجام می‌شوند و متغیرها مقادیر جدید را به خود می‌گیرند.

جداسازی مراحل ۲ و ۳ رفتار قطعی و تکرارپذیر را تضمین می‌کند.

StateCharts

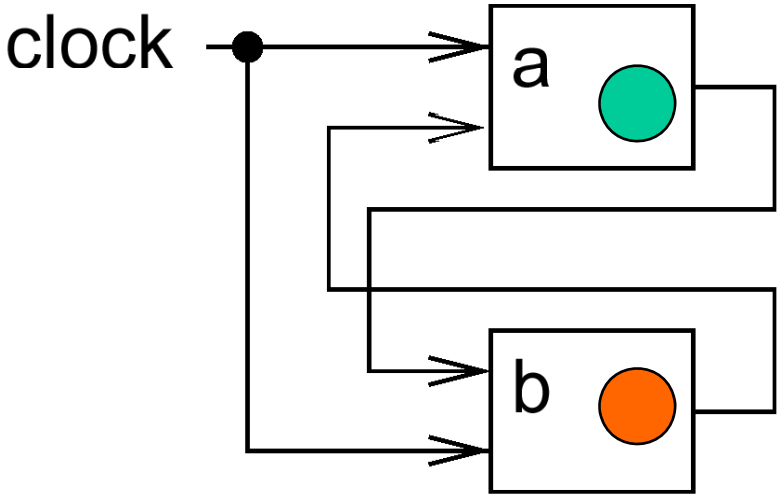
مثال



In phase 2, variables **a** and **b** are assigned to temporary variables.
In phase 3, these are assigned to **a** and **b**.
As a result, variables **a** and **b** are swapped.

In a single phase environment, executing the left state first would assign the old value of **b** (=0) to **a** and **b**.
Executing the right state first would assign the old value of **a** (=1) to **a** and **b**. The execution would be non-deterministic.

StateCharts
انعکاس مدل سخت‌افزار کلاک‌دار

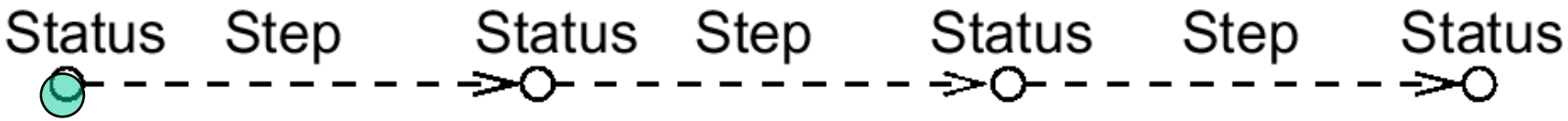


در سیستم‌های سخت‌افزاری واقعی کلاک‌دار (همگام)، هر دو ثبات به درستی مبادله swap می‌شوند.

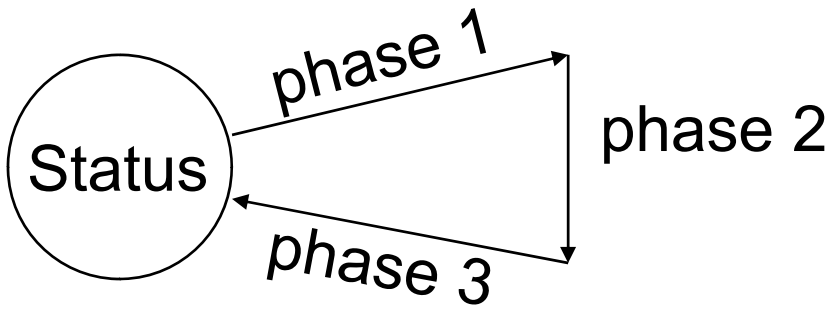
برخی از تفکیک فازها در زبان‌های دیگر به خوبی یافت می‌شود، خصوصاً آنهایی که قصد مدل‌سازی سخت‌افزار را دارند.

StateCharts
گام‌ها

اجرای یک مدل StateChart از دنباله‌ای از جفت‌های (status, step) تشکیل می‌شود.



Status = values of all variables + set of events + current time
Step = execution of the three phases (StateMate semantics)



StateCharts
مکانیزم پخش همگانی

مقادیر متغیرها، برای همه‌ی بخش‌های مدل StateCharts قابل مشاهده است.

مقادیر جدید در مرحله‌ی ۳ گام جاری مؤثر می‌شوند و بوسیله‌ی همه‌ی بخش‌های مدل در گام زیر به دست می‌آیند:

- StateCharts به طور ضمنی یک مکانیزم **پخش همگانی (broadcast)** را برای متغیرها فرض می‌کند.
- StateCharts برای سیستم‌های کنترل محلی مناسب است،
  اما برای کاربردهای توزیع‌شده که بهنگام سازی متغیرها ممکن است مقداری زمان بگیرد مناسب نیست.

StateCharts
دوران زندگی رویدادها

رویدادها زنده هستند تا وقتی که گامی که در آن تولید شده‌اند، طی شود. (رویدادهای one-shot)

StateCharts
ارزیابی StateCharts

مزایا:

- سلسله‌مراتب امکان تودرتوسازی دلخواه ابرحالت‌های AND و OR را فراهم می‌کند.
- تعداد زیادی ابزار شبیه‌سازی تجاری وجود دارد:

  StateMate، StateFlow، BetterState، . . .
- ابزارهایی برای ترجمه‌ی آنها به کد C یا VHDL وجود دارد،
  بنابراین امکان پیاده‌سازی نرم‌افزاری یا سخت‌افزاری آنها وجود دارد.

StateCharts
ارزیابی StateCharts (ادامه)

معایب:

- برنامه‌های C تولیدشده اغلب ناکارامد هستند.
- برای کاربردهای توزیع‌شده مناسب نیست.
- ساختارهای برنامه‌سازی در آن وجود ندارد.
- رفتارهای غیرتابعی را نمی‌توان در آن توصیف کرد.
- شیءِ گرایی را پشتیبانی نمی‌کند.
- سلسله‌مراتب ساختاری را نمی‌توان در آن توصیف کرد.

توسعه‌ها:

- چارت‌های پیمانه‌ای برای توصیف سلسله‌مراتب ساختاری.

خلاصه

Requirements for specification languages

- Hierarchy
- Timing behavior
- State-oriented behavior
- Concurrency
- Synchronization & communication, …

Models of computation

StateCharts

- AND-states
- OR-states
- Timer
- Broadcast
- Semantics
- …