

PA160: Net-Centric Computing II.

Specification and Verification of Network Protocols

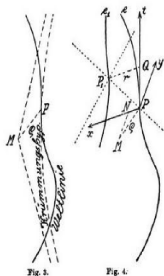
Vojtěch Řehák

Faculty of Informatics Masaryk University

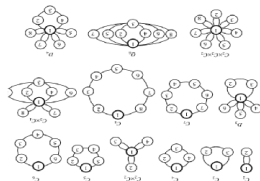
Spring 2011

Theory

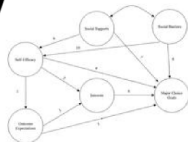
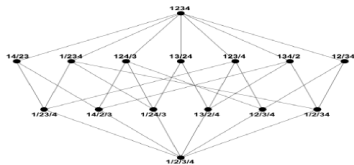
Theory Definition and Usage



$$\mathcal{L} = m \sqrt{g_{\mu\nu} \frac{dx^\mu}{d\tau} \frac{dx^\nu}{d\tau}}$$



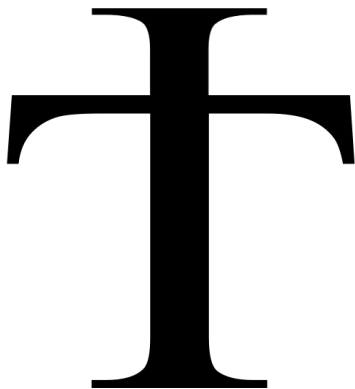
Theory



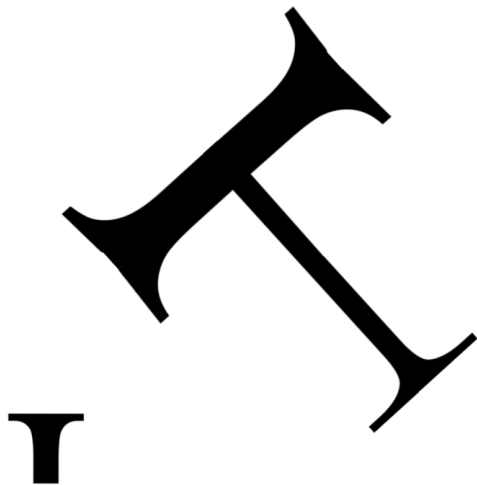
$$\int_a^b f(x) dx = F(b) - F(a)$$

T

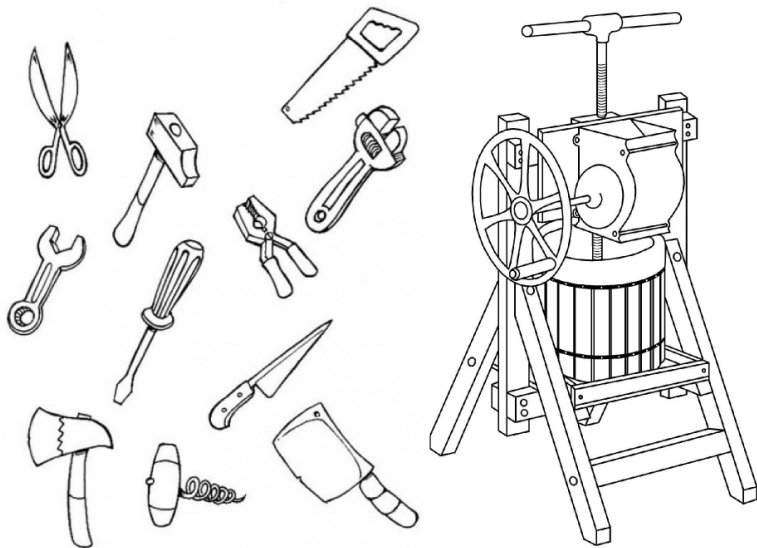
Theory Definition and Usage



Theory Definition and Usage



Theory Definition and Usage



Theoretical Results as Tools for Users

Formal Models - what are they good for

The basic concept is a *model* of system (i.e. the object we work with).


thought handling

- individual approach (intra-brain) - to grasp it
- documentation (inter-brain) - to pass it on


automatic/computer processing (comparing model to specification)


- testing
- simulation
- symbolic execution
- static analysis
- model checking
- equivalence checking
- theorem proving

natural language **vs.** formal language

freedom in writing 


human resources 

 restrictions in writing


 automatic methods


Formal Models in Specification

natural language **vs.** formal language

freedom in writing 

human resources 


 restrictions in writing

 automatic methods


nothing \leq text \leq structured text \leq text with formal “pictures” \leq
formal description with informal comments \leq fully formal description


Formal Models in Specification

natural language **vs.** formal language

freedom in writing 

human resources 


 restrictions in writing

 automatic methods

nothing \leq text \leq structured text \leq text with formal “pictures” \leq
formal description with informal comments \leq fully formal description


Goal: Find an appropriate level of abstraction and keep it.

natural language **vs.** formal language

freedom in writing 

human resources 

 restrictions in writing

 automatic methods

nothing \leq text \leq structured text \leq text with formal “pictures” \leq
formal description with informal comments \leq fully formal description

Goal: Find an appropriate level of abstraction and keep it.

“What will be the model used for?”

Map - Abstraction Example



Find Pardubice or directions from Brno to Liberec.

source:www.mapy.cz

Map - Abstraction Example



Find Pardubice or directions from Brno to Liberec.

source:www.mapy.cz

“Model has to suit its **purpose!**”

Only relevant information are presented; no more, no less.

Outline

Models we will talk about:

- Specification and Description Language (SDL)
- Message Sequence Charts (MSC)
- Petri nets

What they can be used for?

- modelling
- specification
- analysis
- simulation
- testing
- partial implementation

Distributed Systems – Key characteristics (Déjà vu)

Key Characteristics of Distributed Systems:

- **Autonomy** – there are several autonomous computational entities, each of which has its own local memory
- **Heterogeneity** – the entities may differ in many ways
 - computer HW (different data types' representation), network interconnection, operating systems (different APIs), programming languages (different data structures), implementations by different developers, etc.
- **Concurrency** – concurrent (distributed) program execution and resource access
- **No global clock** – programs (distributed components) coordinate their actions by exchanging messages
 - message communication can be affected by delays, can suffer from variety of failures, and is vulnerable to security attacks
- **Independent failures** – each component of the system can fail independently, leaving the others still running (and possibly not informed about the failure)
 - How to know/differ the states when a network has failed or became unusually slow?
 - How to know if a remote server crashed immediately?

Specification Description Language (SDL)

Specification Description Language (SDL)

Specification Description Language (SDL)

international standard of the ITU-T, Z.100

- 1972 - Establishment of a working group for SDL
- 1976 - first version of Z.100 recommendation
- ...
- 2007 - current version of Z.100 recommendation
- all documents of the current version:
 - Z.100 - Specification and Description Language (SDL)
 - Z.100 Annex F1 - SDL formal definition: General overview
 - Z.100 Annex F2 - SDL formal definition: Static semantics
 - Z.100 Annex F3 - SDL formal definition: Dynamic semantics
 - Z.100 Supplement 1 - SDL+ methodology: Use of MSC and SDL
 - Z.Imp100 - SDL implementer's guide
 - Z.104 - Encoding of SDL data
 - Z.105 - SDL combined with ASN.1 modules
 - Z.106 - Common interchange format for SDL
 - Z.109 - SDL-2000 combined with UML

SDL - Specification Description Language

An **object oriented** languages for specification of applications that are

- heterogeneous,
- distributed (concurrent),
- interactive (event-driven, discrete signals), and
- real-time dependent (with delays, timeouts).

Describes

- structure (distributed components of the system),
- behaviour (instructions within the components), and
- data

of distributed systems in **real-time** environments.

What SDL is good for?

SDL is designed for unambiguous **specification** of requirements and **description** of implementation of the normative requirements of **telecommunication protocol** standards.

For computer based tools to improve the process of

- specification (create, maintain, and analyze), and
- implementation (automatic code generation).

What SDL is good for?

SDL is designed for unambiguous **specification** of requirements and **description** of implementation of the normative requirements of **telecommunication protocol** standards.

For computer based tools to improve the process of

- specification (create, maintain, and analyze), and
- implementation (automatic code generation).

What SDL is NOT good for?

high level system description (what the system serves for), demonstration of good or wrong behaviour, test trace specification, implementation details (primitives for communication, detailed data manipulation), etc.

SDL - representations

Three representations:

SDL/GR graphical representation (human readable)

SDL/PR textual phrase representation (machine readable)

SDL/CIF common interchange format (SDL/PR with graphical information)

SDL - representations

Three representations:

SDL/GR graphical representation (human readable)

SDL/PR textual phrase representation (machine readable)

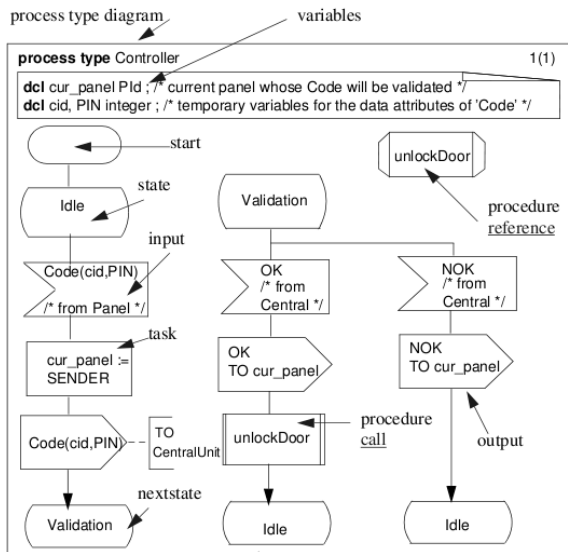
SDL/CIF common interchange format (SDL/PR with graphical information)

In what follows, we focus on the graphical representation (SDL-GR).

Basic SDL components

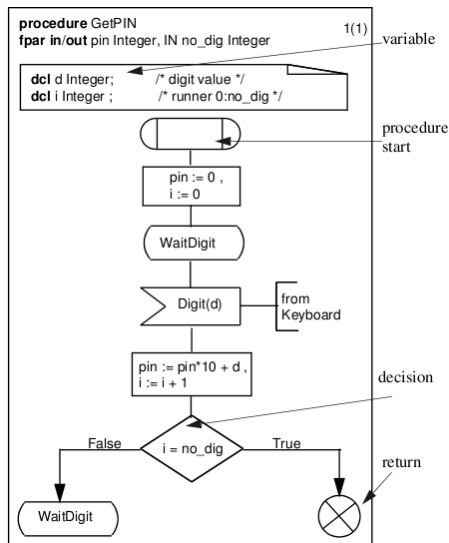
- system and blocks (structure)
- processes and procedures (behaviour)

SDL/GR - Process



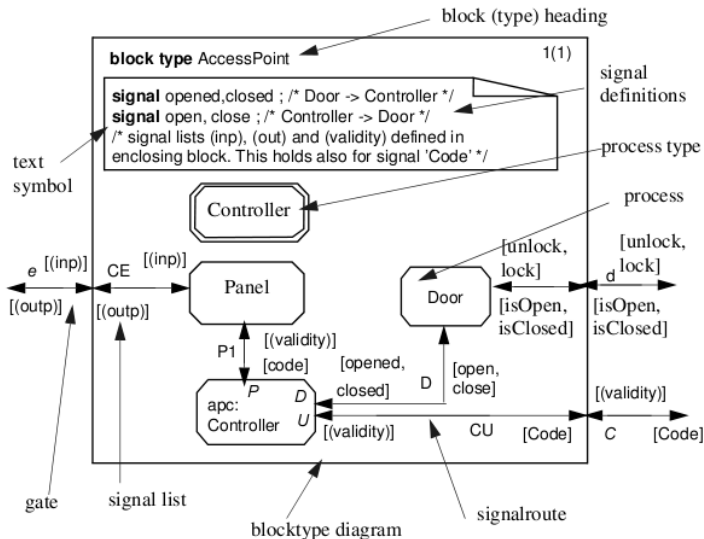
source: TIME Electronic Textbook v4.0

SDL/GR - Procedure

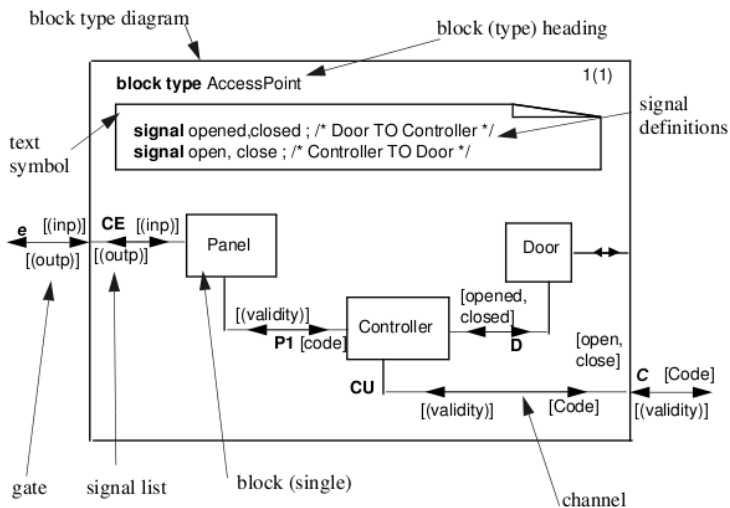


source: TImE Electronic Textbook v4.0

SDL/GR - Block

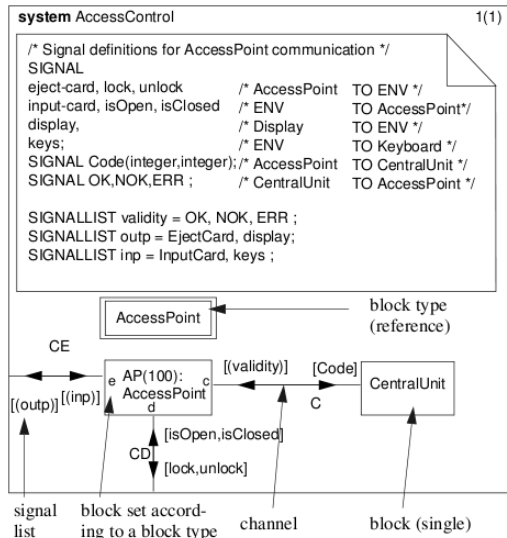


SDL/GR - Block with block structure



source: TImE Electronic Textbook v4.0

SDL/GR - System (the top most block)



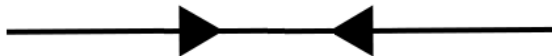
source: TImE Electronic Textbook v4.0

SDL/GR - Channels

Nondelaying channels for “immediate” message delivery (e.g., between processes within a computer). Nondelaying channels



Delaying channels for “time consuming” message delivery (e.g., between dislocated blocks).



Channels can also be one-directional.

Summary of SDL Basics

System is the top most block surrounded by environment.

Block consists of blocks or processes that are connected by channels.

express the hierarchical structure of the system.

names are references to other objects.

Process sends and receives messages.

stay in states.

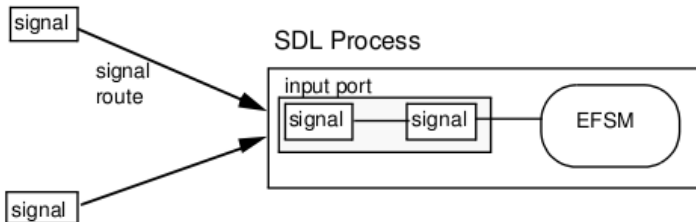
can call procedures.

Procedure is a subroutine that can finish.

does not return any value (only in variables or sent messages).

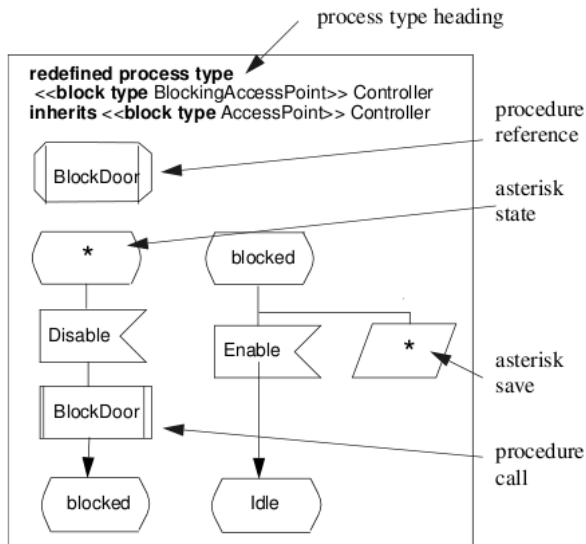
Message Exchange

- one input buffer for a process
- FIFO behaviour
- no priority queues
- signal which is unspecified in the current state is **discarded**



source: TIME Electronic Textbook v4.0

Asterisk Save, Asterisk State, and Dash State



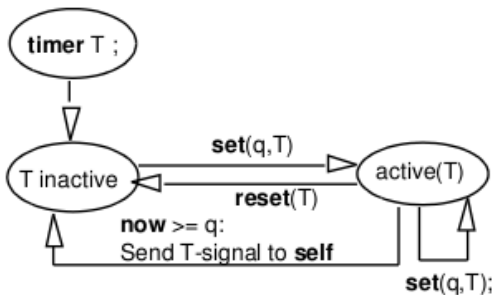
source: TImE Electronic Textbook v4.0

Timer Construction

TIMER door_timeout ;

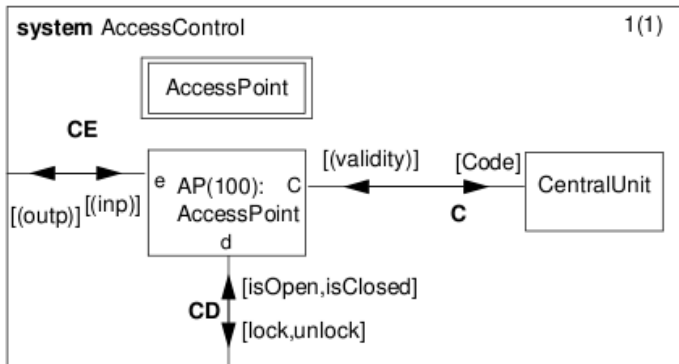
set
(now +10,
door_timeout)

door_timeout



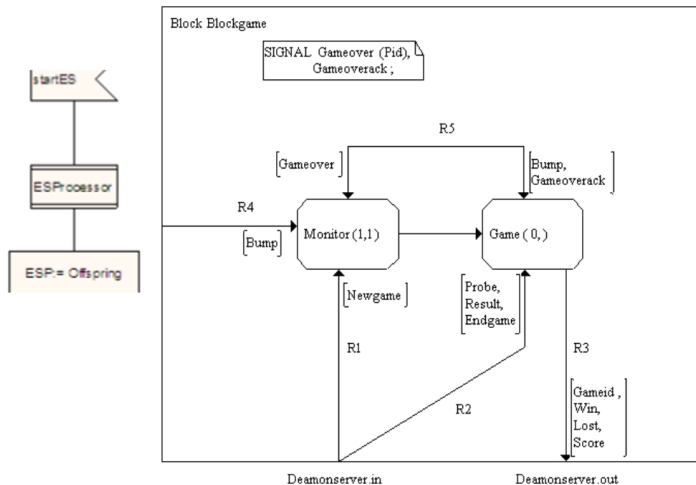
source: TImE Electronic Textbook v4.0

Multiple Instances of a Block



source: TIme Electronic Textbook v4.0

Multiple Instances of a Process

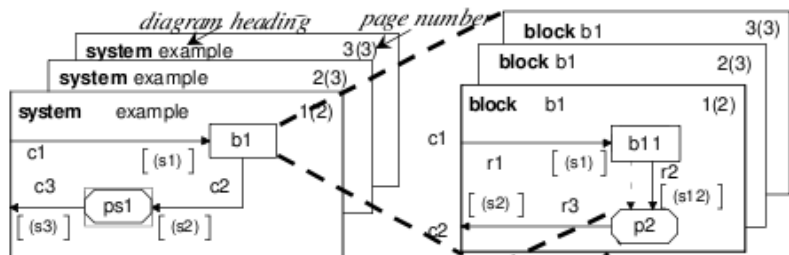


Additional SDL Constructs

- **Asterisk save, asterisk state, and dash state**
- **Timer** construction
- **Multiple block instances** (no dynamic creation)
- **Multiple process instances** (with dynamic creation and limit)

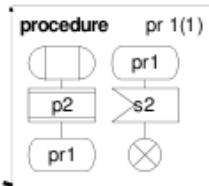
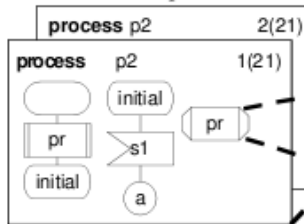
- **Packages** collections of related types and definitions (library)
- **Subtypes**
- **Virtual processes**
- **Process type redefinition** and **finalization**
- **Inherited blocks**

SDL - Overview



Systems and blocks can contain blocks and/or processes.

Processes contain behaviour and cannot contain blocks.



IBM Rational

- from tools of Telelogic (SDT, Geode, Tau)
- drawing, import, export
- automatic implementation in C++
- simulation support

SanDriLa SDL

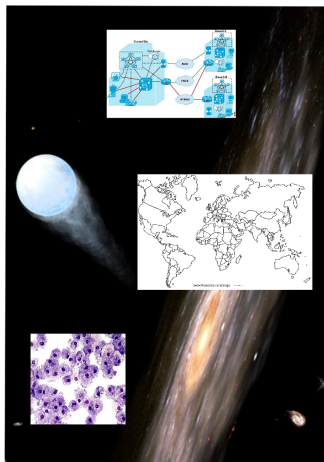
- MS Visio stencil
- drawing, import, export
- analyses of states in process diagrams
- open for addons

Cinderella SDL

- modelling, import, export
- analyses and simulation

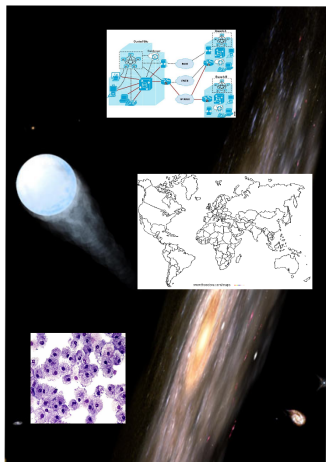
“What is the problem of distributed systems?”

Distributed Systems

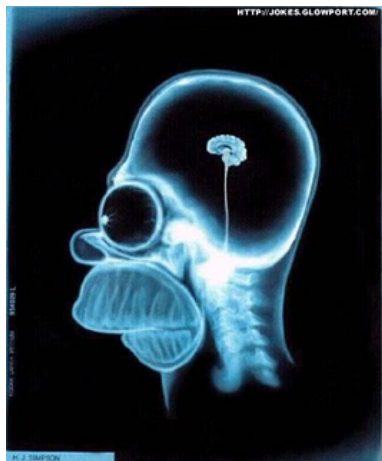


World is distributed

Distributed Systems



World is distributed



Human way of thinking is sequential

SDL

Specification Description
Language

ITU-T Z.100

MSC

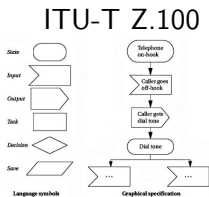
Message Sequence
Chart

ITU-T Z.120

Distributed vs. Local

SDL

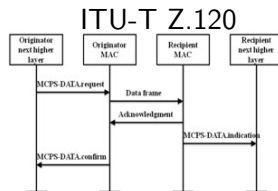
Specification Description
Language



models of components

MSC

Message Sequence
Chart



communication model

Message Sequence Chart (MSC)

Message Sequence Chart (MSC)

Message Sequence Chart (MSC)

international standard of the ITU-T, Z.120

- 1993 - first version of Z.120 recommendation
- ...
- 2011 - current version of Z.120 recommendation
- all documents of the current version:
 - Z.120 - Message Sequence Chart (MSC)
 - Z.120 Annex B - Formal semantics of message sequence charts
 - Z.121 - Specification and Description Language (SDL) data binding to Message Sequence Charts (MSC)

Message Sequence Chart (MSC)

A **trace language** for the specification and description of the **communication behaviour** by means of **message exchange**.

Describes

- communicating processes,
- communication traces,
- message order,
- time information (timeouts, constraints),
- high-level form for set of traces.

What MSC is good for?

Both **human** and computer readable formalizm for:

- basic behaviour demonstration (use cases),
- high level system behaviour description,
- test case specification, and
- (test) log visualization.

What MSC is good for?

Both **human** and computer readable formalizm for:

- basic behaviour demonstration (use cases),
- high level system behaviour description,
- test case specification, and
- (test) log visualization.

What MSC is NOT good for?

detailed specification (before implementation), hierarchical structure of communicating entities, implementation details (primitives for communication, detailed data manipulation), etc.

MSC and SDL in Workflow

- typical/optimal communication sequences (MSC)
- error sequences (MSC)
- optionally - full specification in (HMSC)
- distributed specification (SDL)

MSC and SDL in Workflow

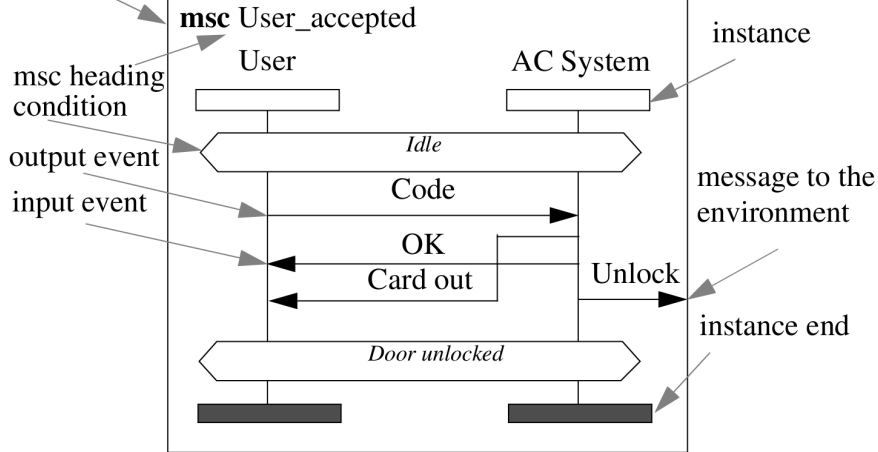
- typical/optimal communication sequences (MSC)
- error sequences (MSC)
- optionally - full specification in (HMSC)
- distributed specification (SDL)

Formal model benefits

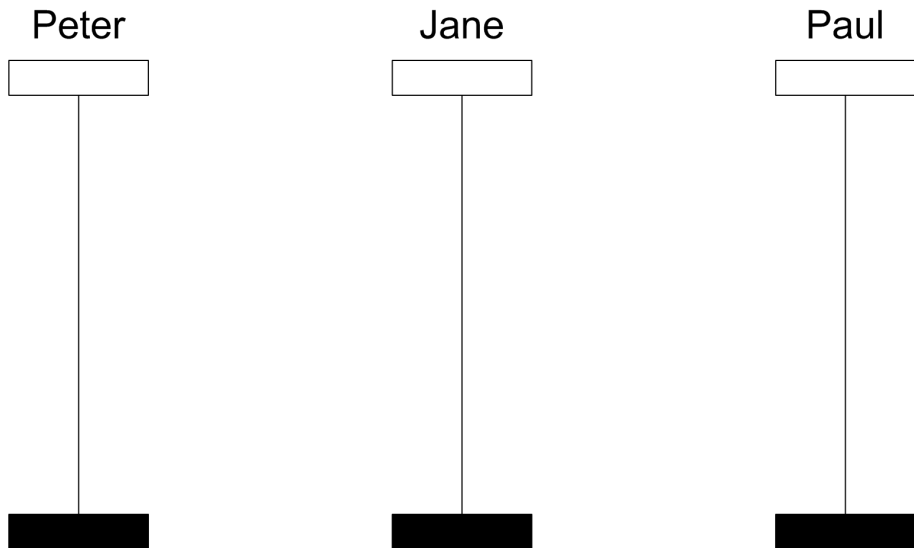
- (H)MSC to SDL transformation (realization)
- SDL to source code transformation (implementation)
- MSC to test case transformation
- simulation to MSC transformation (membership checking)
- ...

Message Sequence Chart (MSC)

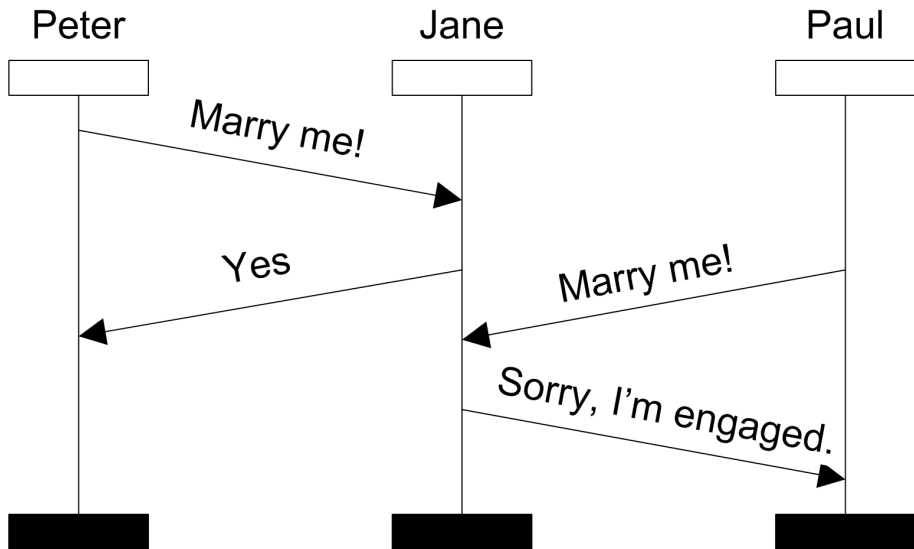
msc diagram



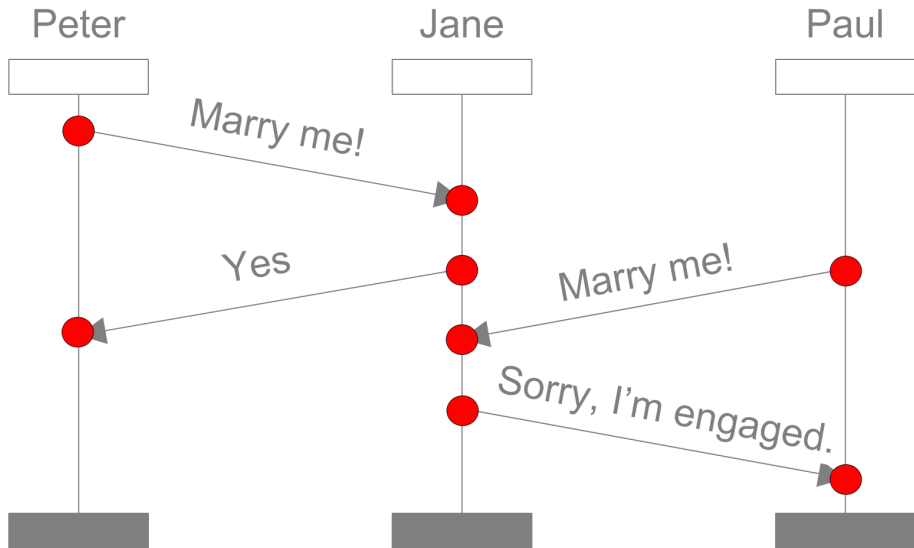
Message Sequence Chart (MSC) - semantics



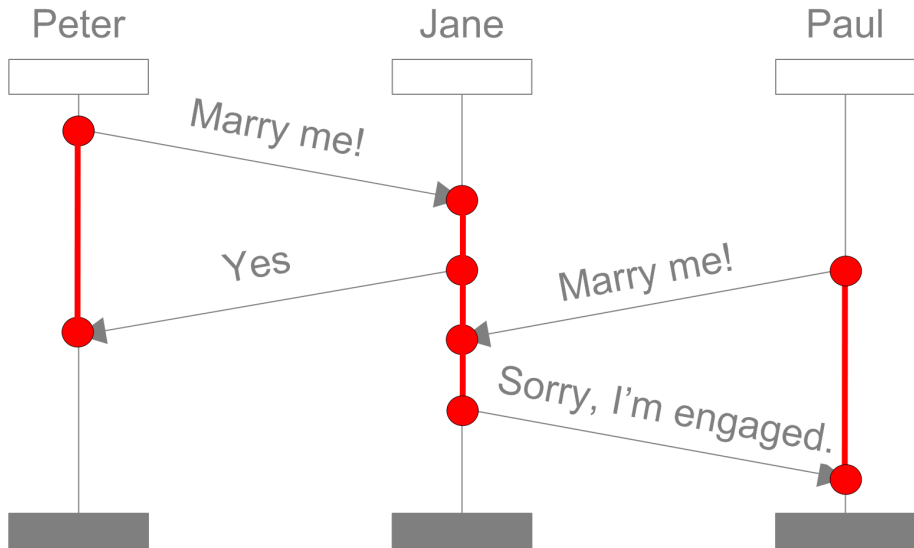
Message Sequence Chart (MSC) - semantics



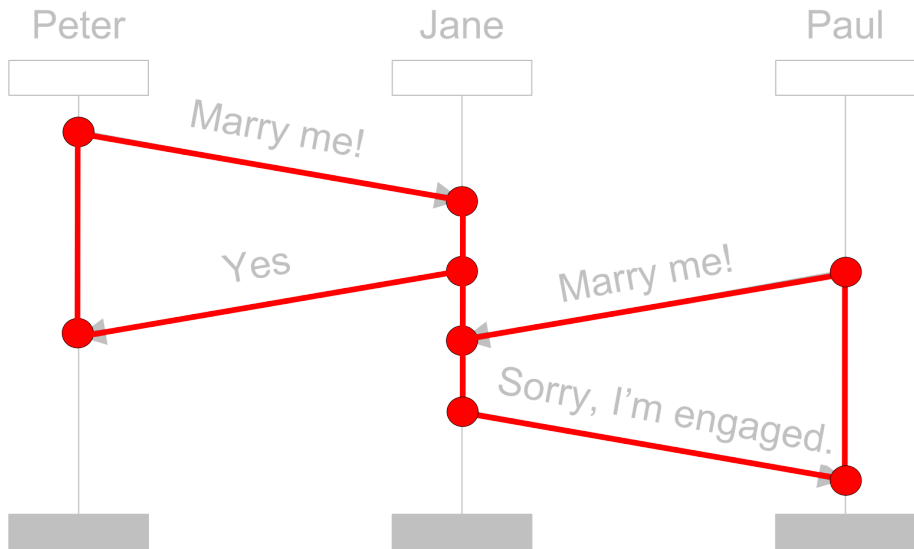
Message Sequence Chart (MSC) - semantics



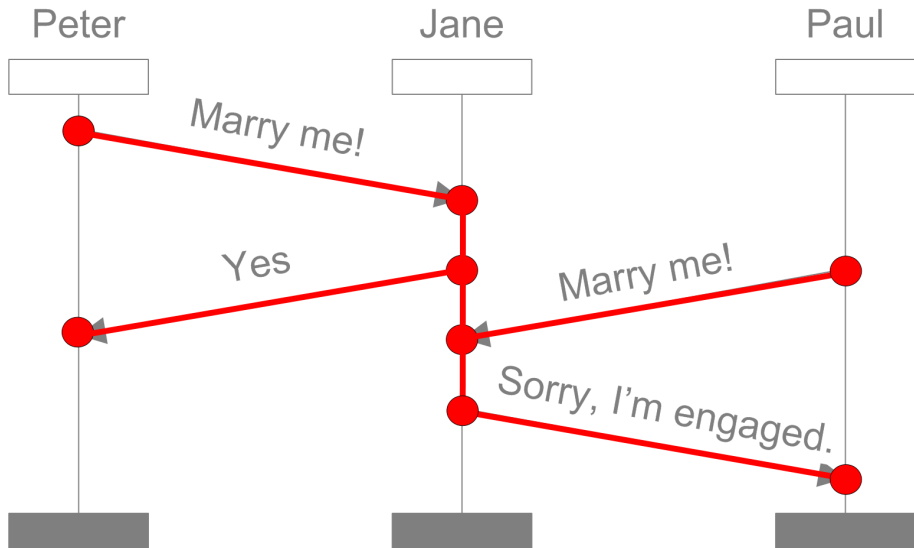
Message Sequence Chart (MSC) - semantics



MSC - Visual Order



MSC - Visual Order - Hasse Diagram



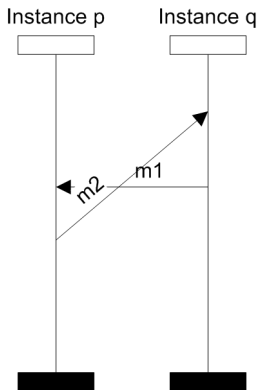
What is an unwanted behaviour/property?

What is an unwanted behaviour/property?

Fundamental problems in the specified model, e.g. an implementation of the model does not exist in the given environment.

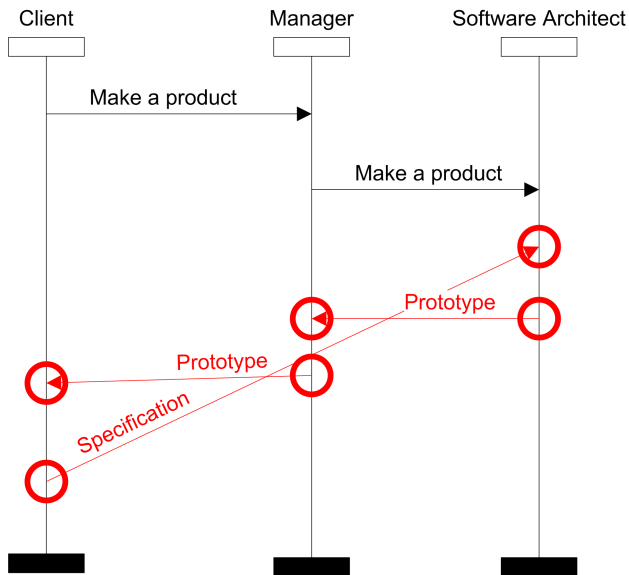
Acyclic/Cyclic property

cyclic dependency among events



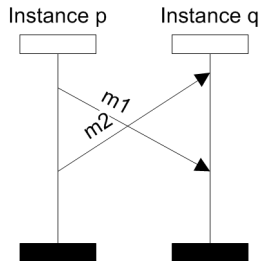
unrealizable in any environment

Acyclic/Cyclic property



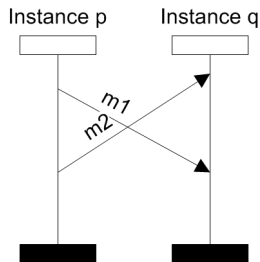
FIFO/non-FIFO property

overleaping messages



FIFO/non-FIFO property

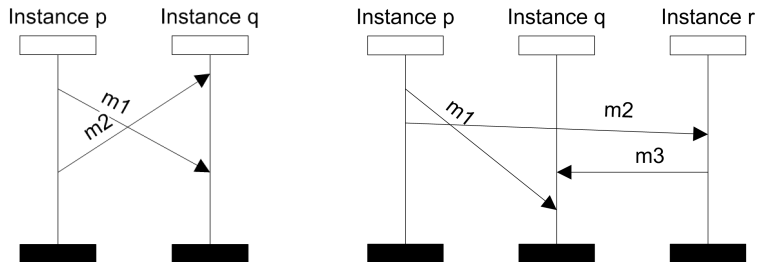
overleaping messages



unrealizable in an environment preserving message order

FIFO/non-FIFO property

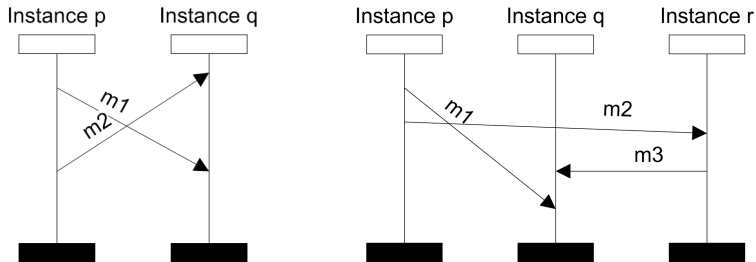
overleaping messages



unrealizable in an environment preserving message order

FIFO/non-FIFO property

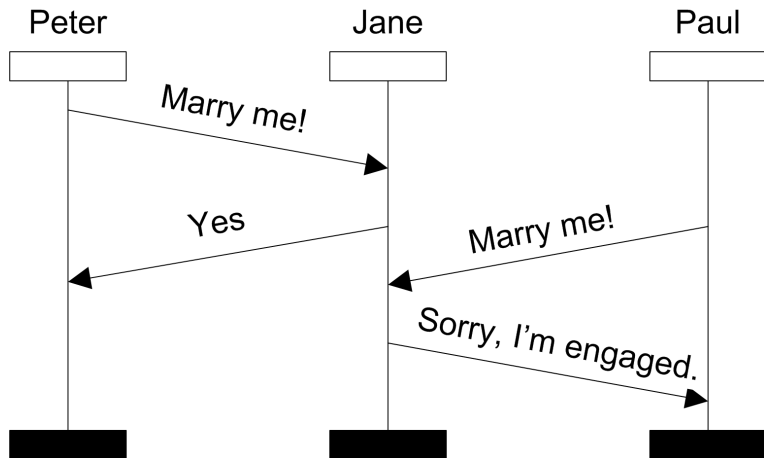
overleaping messages



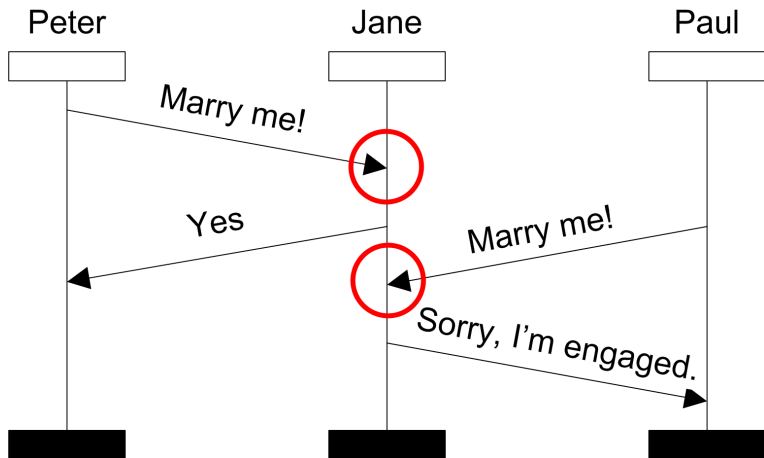
unrealizable in an environment preserving message order

realizable in an environment with P2P channel but unrealizable in case of a global channel

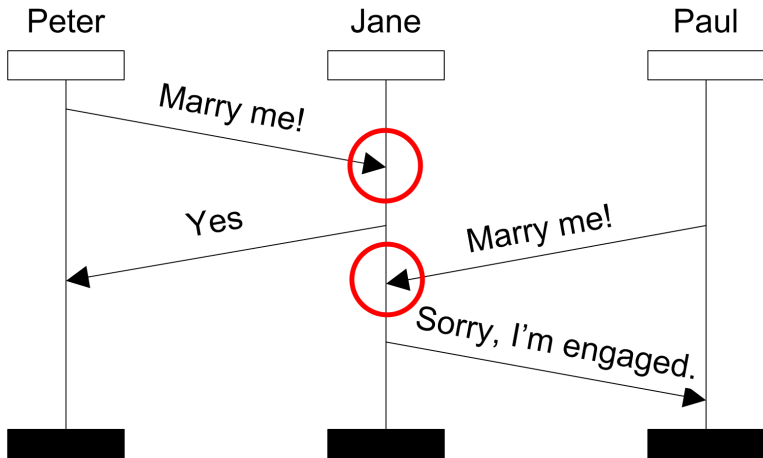
Race Condition



Race Condition



Race Condition



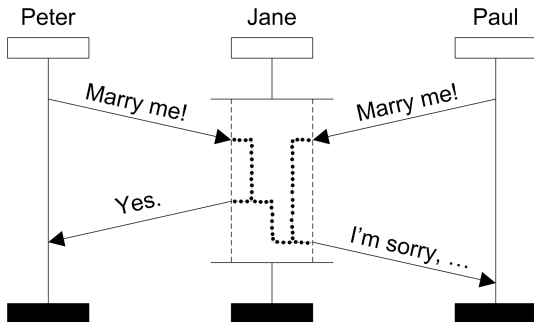
Informally, **race** is when some receive event can come earlier.

Solution #1 - Coregion Construction

Let us demonstrate that some events are not ordered.

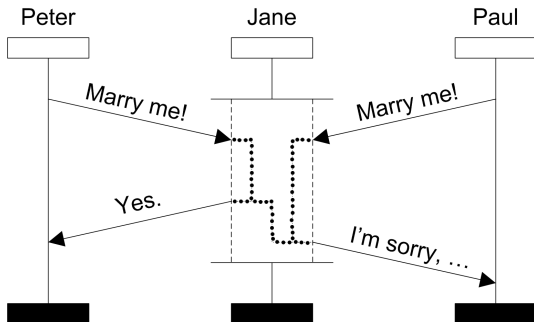
Solution #1 - Coregion Construction

Let us demonstrate that some events are not ordered.



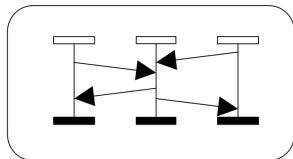
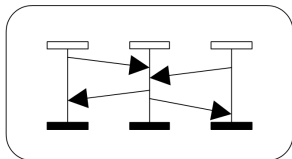
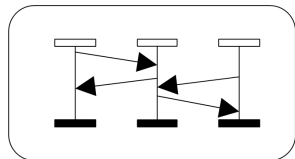
Solution #1 - Coregion Construction

Let us demonstrate that some events are not ordered.

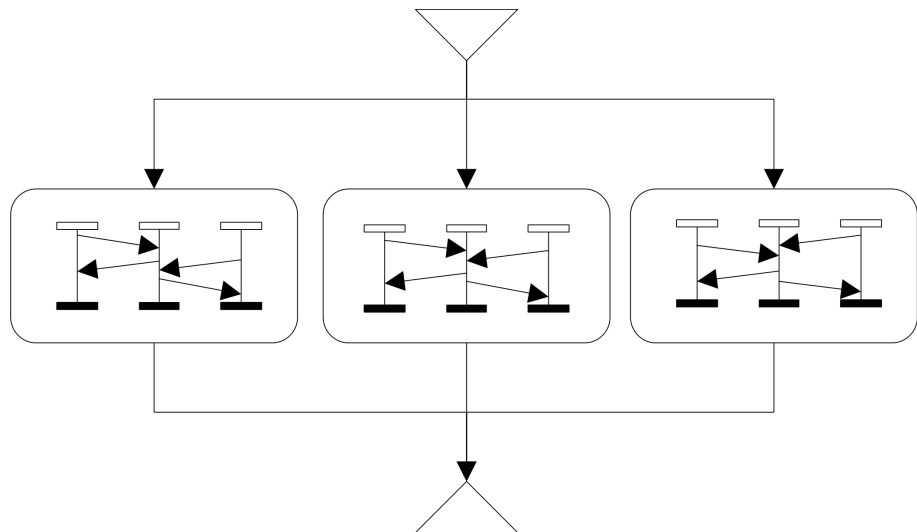


Events in a *coregion* are not order; except of related by *general ordering*.

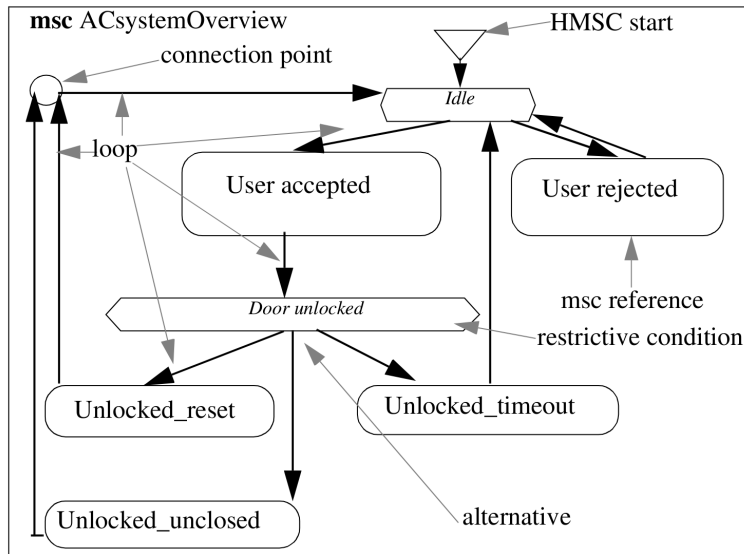
Solution #2 - List/set of all possibilities



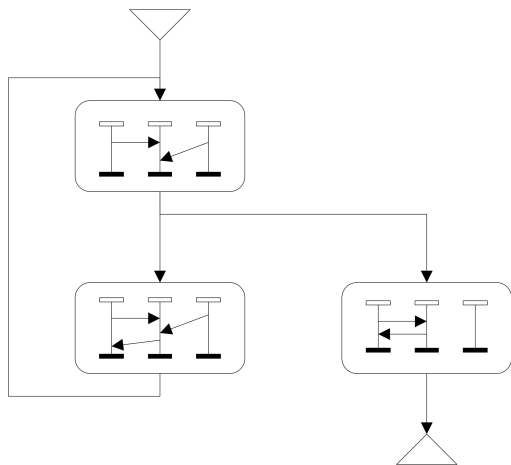
High-Level MSC (HMSC) - ITU-T Z.120



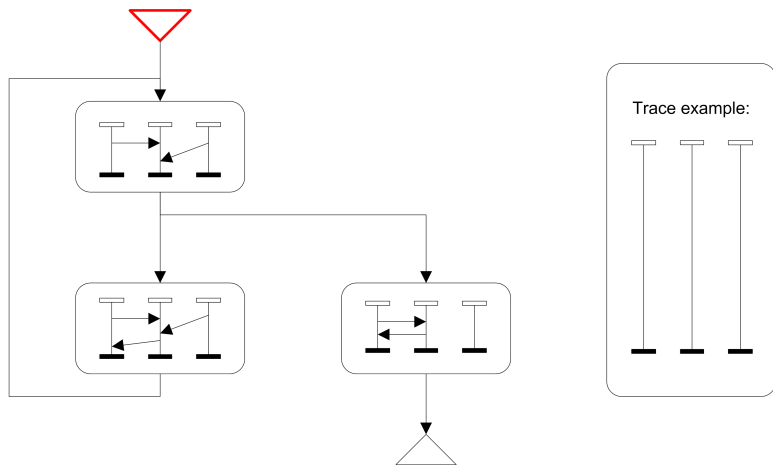
High-Level MSC (HMSC) - ITU-T Z.120



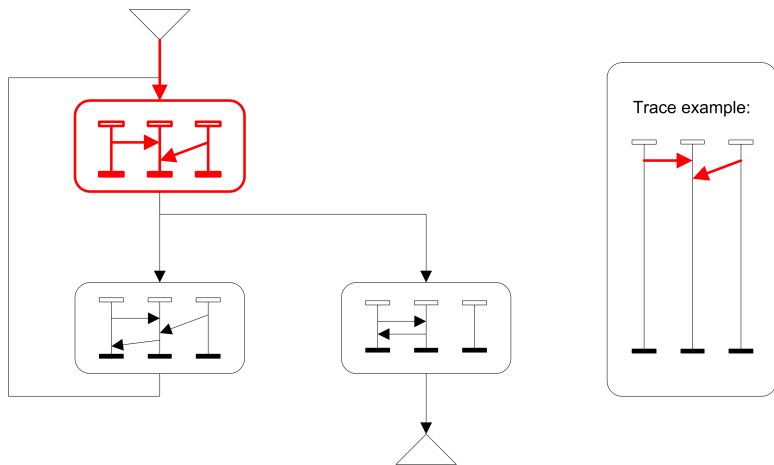
High-Level MSC (HMSC) - ITU-T Z.120



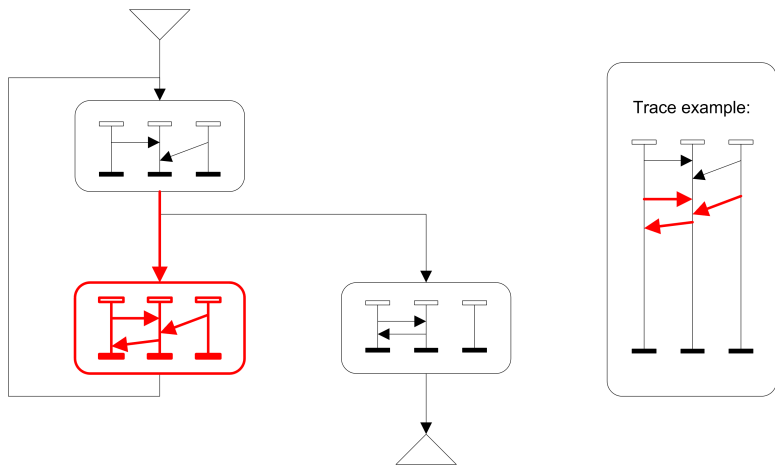
High-Level MSC (HMSC) - ITU-T Z.120



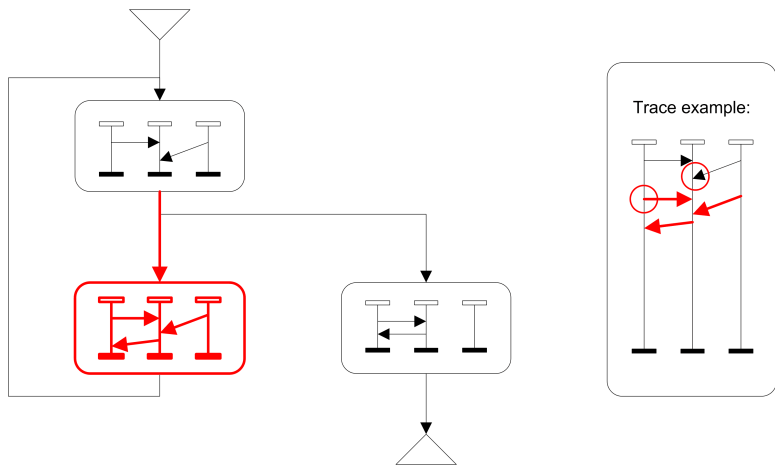
High-Level MSC (HMSC) - ITU-T Z.120



High-Level MSC (HMSC) - ITU-T Z.120

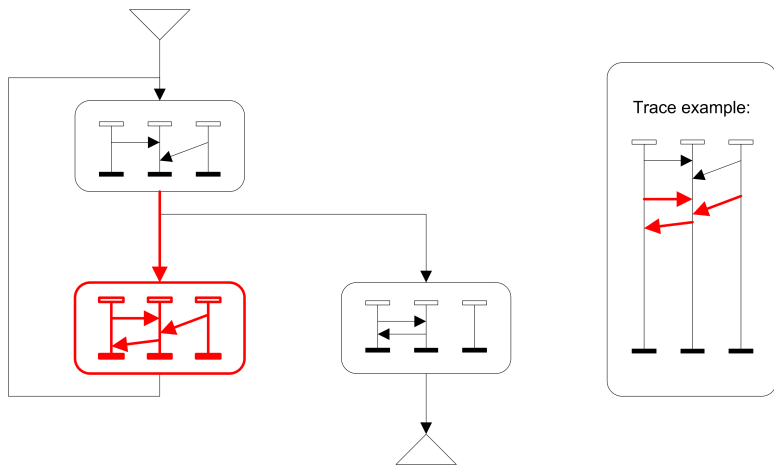


High-Level MSC (HMSC) - ITU-T Z.120

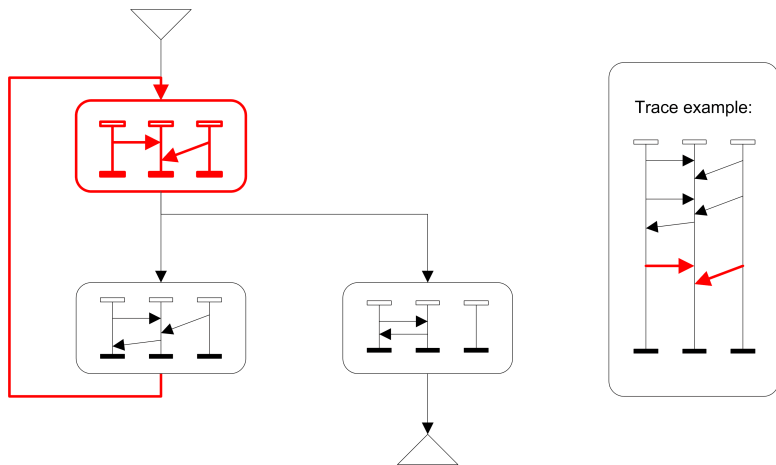


these events are not ordered!

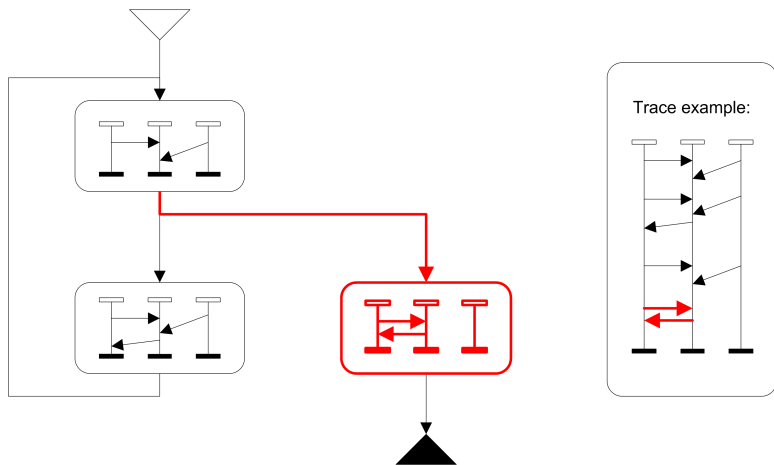
High-Level MSC (HMSC) - ITU-T Z.120



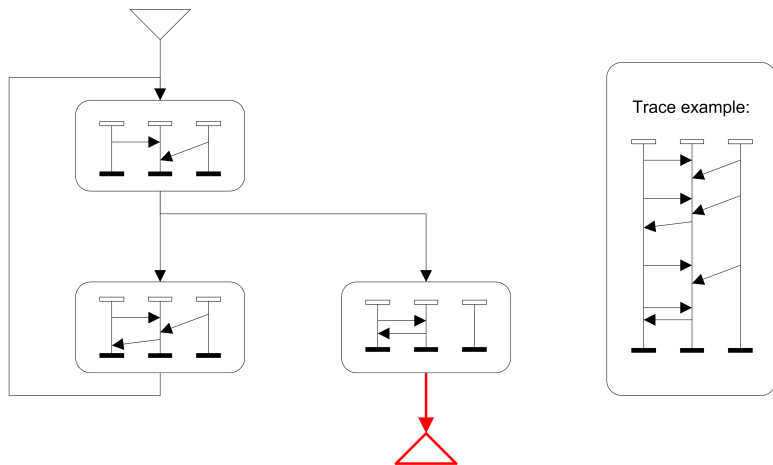
High-Level MSC (HMSC) - ITU-T Z.120



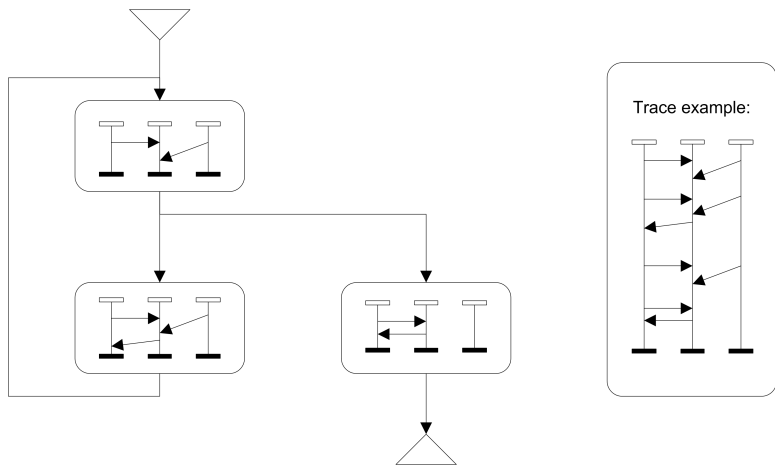
High-Level MSC (HMSC) - ITU-T Z.120



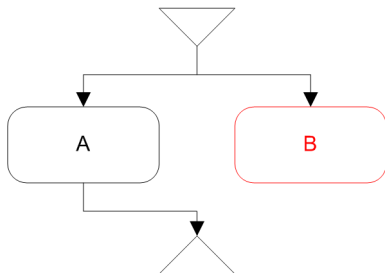
High-Level MSC (HMSC) - ITU-T Z.120



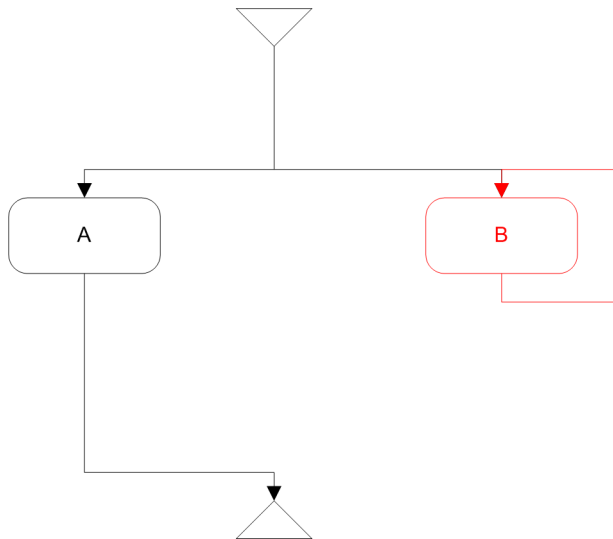
High-Level MSC (HMSC) - ITU-T Z.120



Deadlock Property



Livelock Property

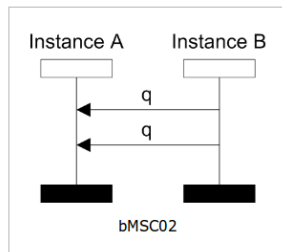
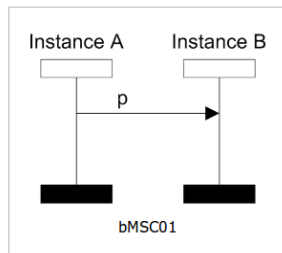
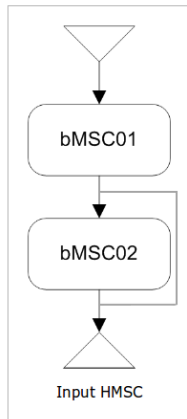
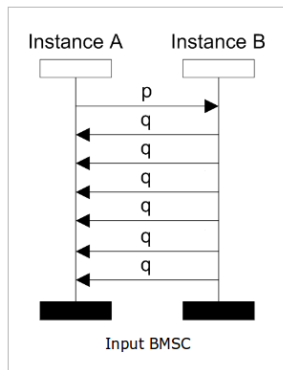


Membership

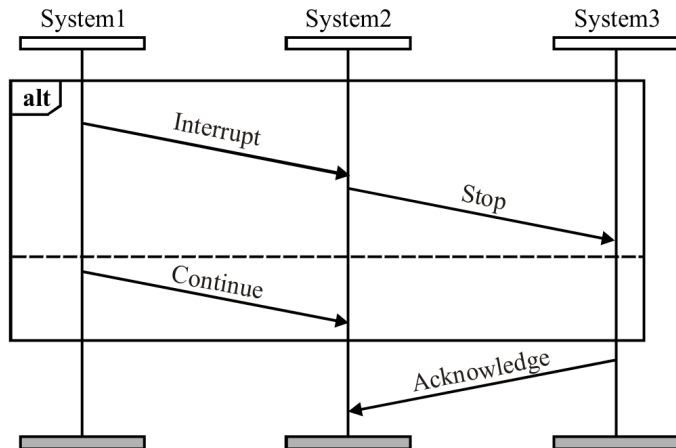
Is a given MSC included in a given HMSC?

Membership

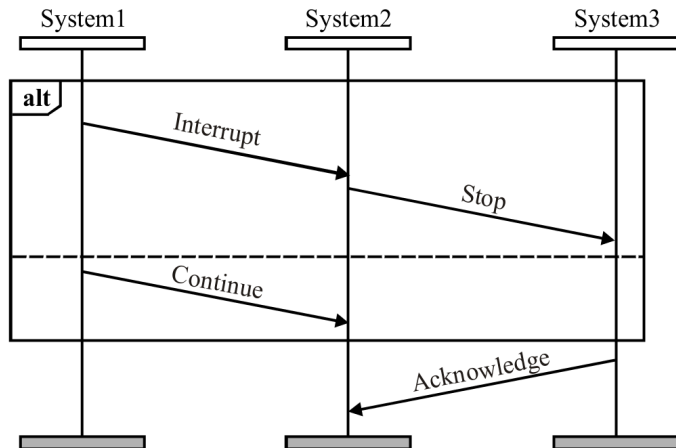
Is a given MSC included in a given HMSC?



Inline Expressions

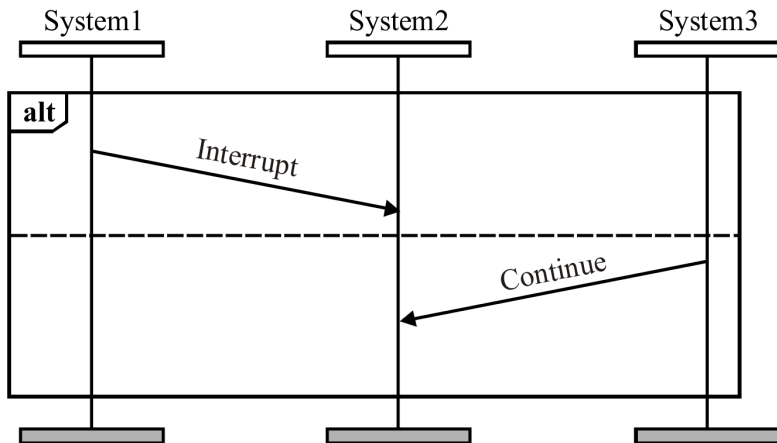


Inline Expressions

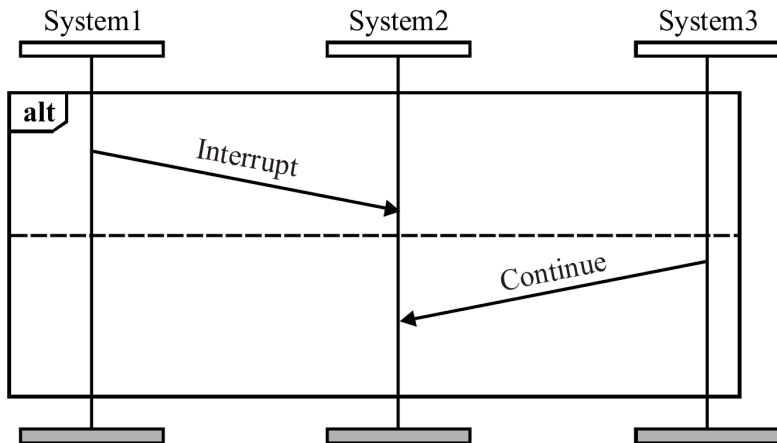


Other inline expression types: **opt**, **loop** $\langle m, n \rangle$, **exc**, **seq**, **par**.

Non-local Choice

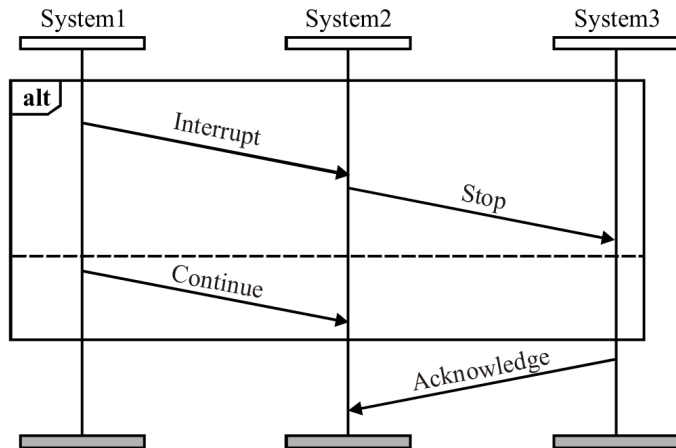


Non-local Choice



System3 does not know which alternative has been chosen.

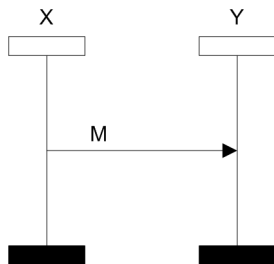
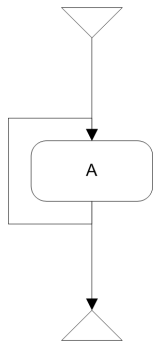
Non-local Choice



System3 does not know which alternative has been chosen.

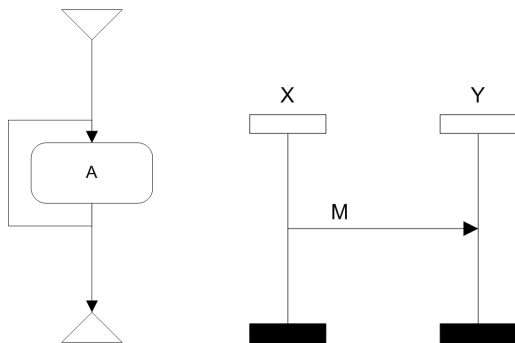
Universal Boundedness

What size of buffer is needed to be sure it will not overflow?



Universal Boundedness

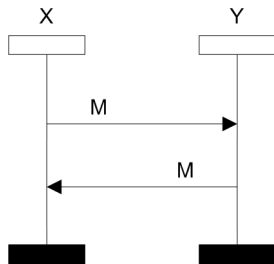
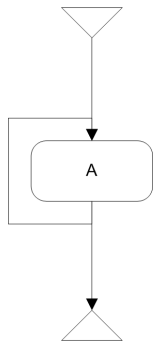
What size of buffer is needed to be sure it will not overflow?



Every finite input buffer of Y can overflow.

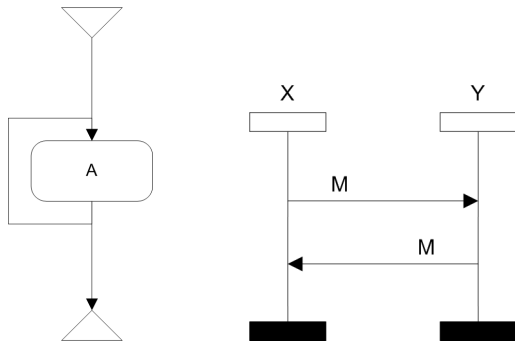
Universal Boundedness

What size of buffer is needed to be sure it will not overflow?



Universal Boundedness

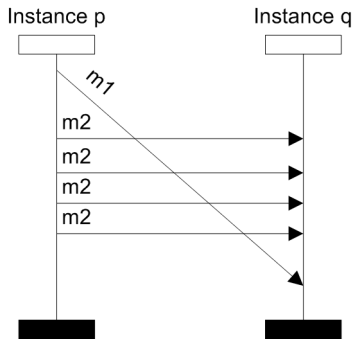
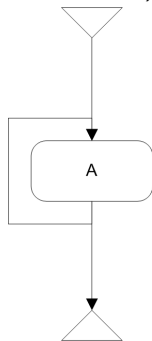
What size of buffer is needed to be sure it will not overflow?



Buffers of size 1 will not overflow.

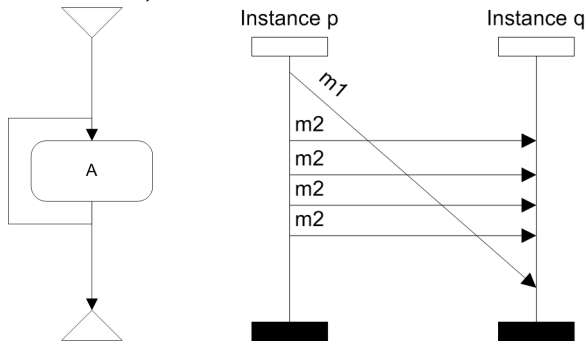
Existential Boundedness

The system deadlocks in case of FIFO channels (and FIFO buffers).
What size of non-FIFO buffers is needed to avoid deadlock (in case of FIFO channels)?



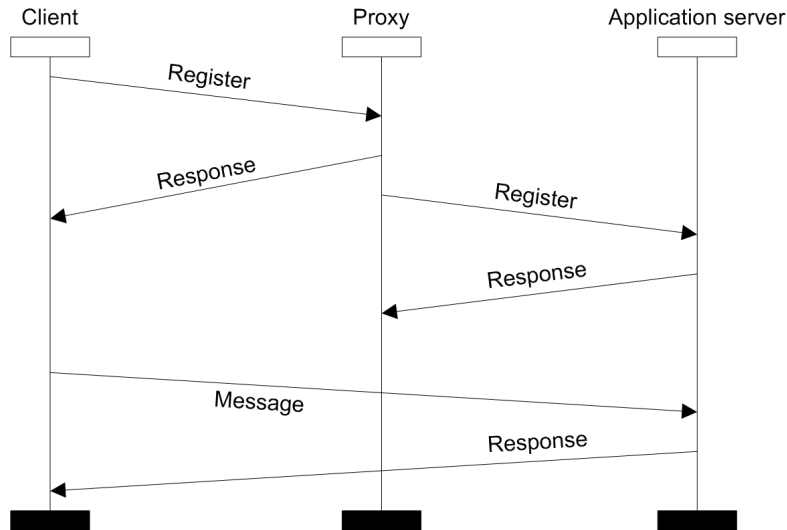
Existential Boundedness

The system deadlocks in case of FIFO channels (and FIFO buffers).
What size of non-FIFO buffers is needed to avoid deadlock (in case of FIFO channels)?

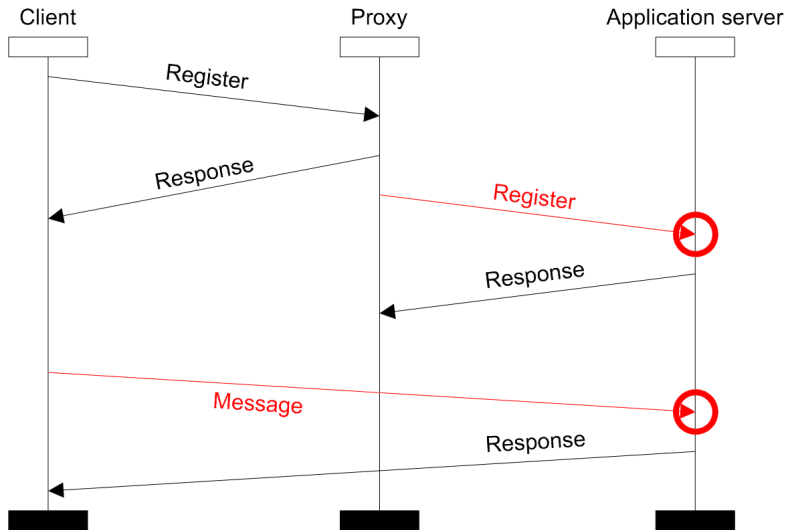


Buffer of size 2 suffices to avoid deadlock.
Or one buffer for each message label (type).

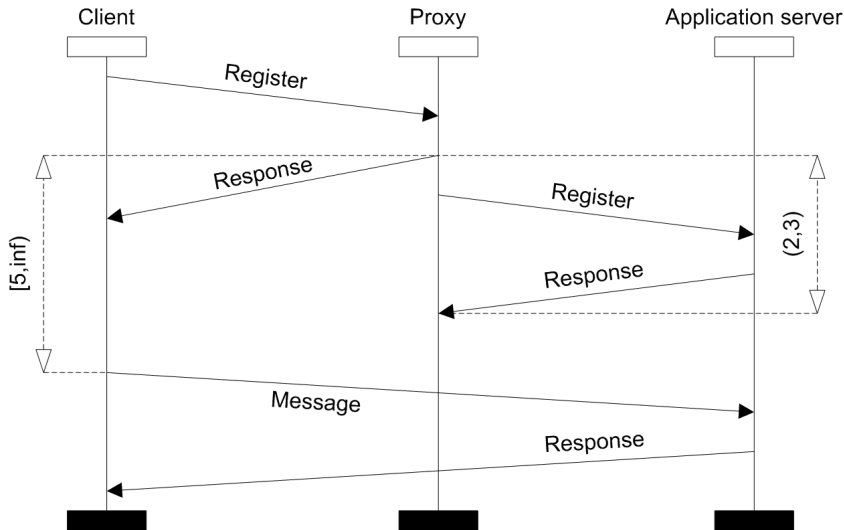
Race Condition - Solution #3 - Time Constraints



Race Condition - Solution #3 - Time Constraints

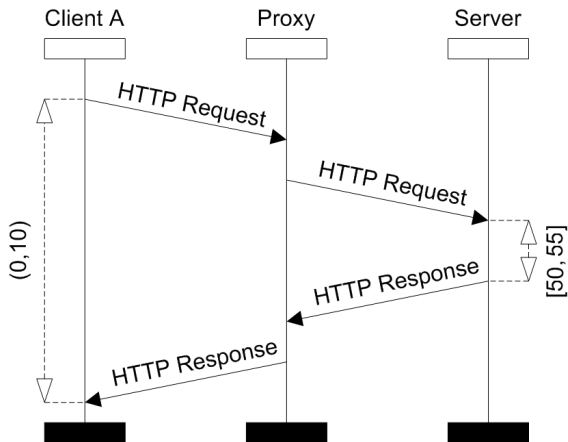


Race Condition - Solution #3 - Time Constraints



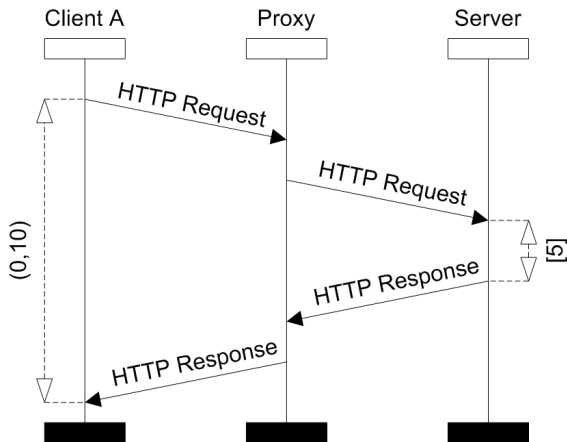
Time Consistency

Are the given time conditions consistent?



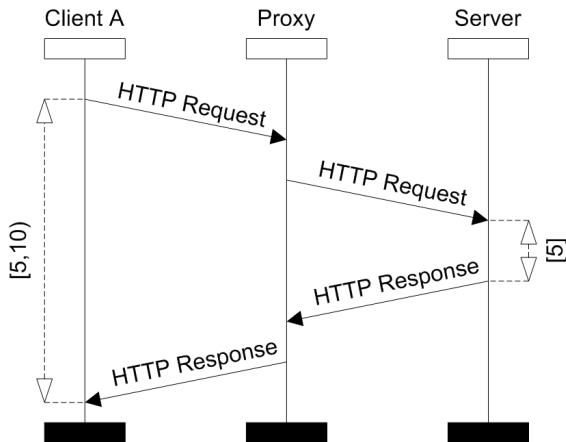
Time Tightening

Some time conditions can be tightened.

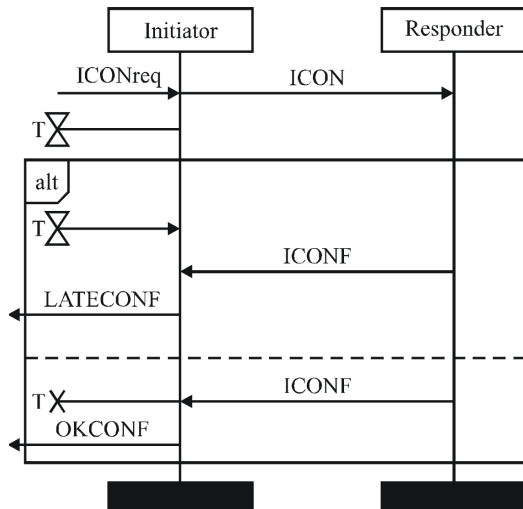


Time Tightening

Some time conditions can be tightened.



Timers



Basic MSC

- instances
- messages
- send events
- receive events
- conditions
- coregions
- general ordering
- inline expressions
- time constraints
- timers

High-level MSC (HMSC)

- start node
- end node
- reference nodes
- connection points
- lines
- conditions
- time constraints

MSC - Properties

- Acyclic property
- FIFO property
- Race Condition

- Deadlock
- Livelock
- Membership
- Nonlocal Choice
- Universal Boundedness
- Existential Boundedness

- Time Race Condition
- Time Consistency
- Tighten Time

IBM Rational, SanDriLa SDL, Cinderella SDL

Sequence Chart Studio (SCStudio)

- MS Visio addon
- drawing, import, export
- checkers for all the mentioned properties

MSCan

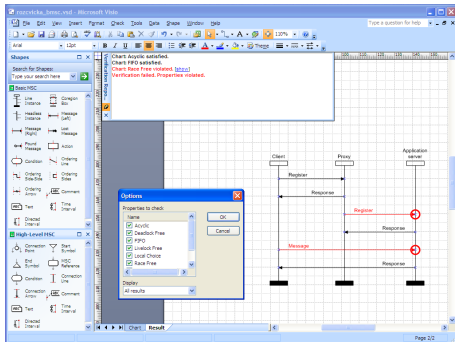
- academic tool
- only textual input
- some checkers

Mesa

- academic tool
- local choice and time checkers

Sequence Chart Studio

MSC drawing and verification tool developed at FI MU.



<http://scstudio.sourceforge.net>

Petri Nets

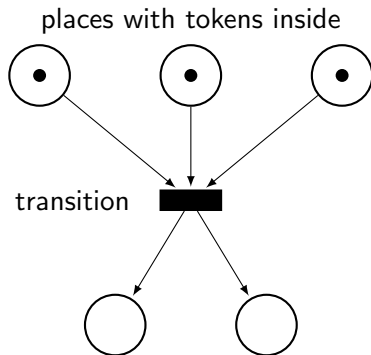
Petri Nets

C. A. Petri: Kommunikation mit automaten, 1962

Basic components:

- places
- transitions
- tokens
- arcs

Marking = configuration
= distribution of tokens
= vector of #s tokens in places



A transition can be fired if there is a token in each of its input places.

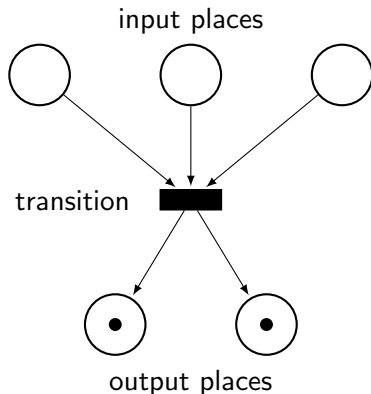
Petri Nets

C. A. Petri: Kommunikation mit automaten, 1962

Basic components:

- places
- transitions
- tokens
- arcs

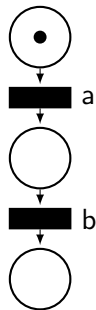
Marking = configuration
= distribution of tokens
= vector of #s tokens in places



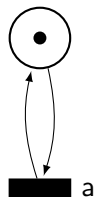
Tokens from input places are removed and new tokens are added into the output places of the fired transition.

Basic Constructions

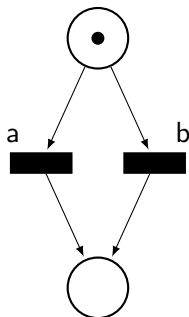
Sequential execution



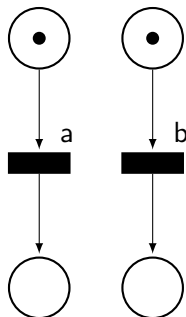
Iteration



Alternative

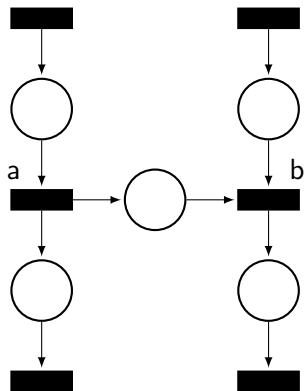


Parallel execution

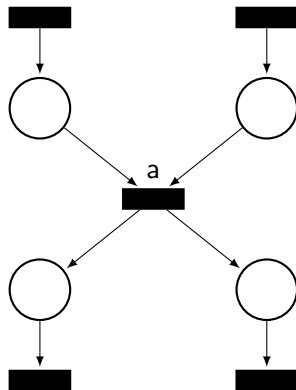


Basic Constructions

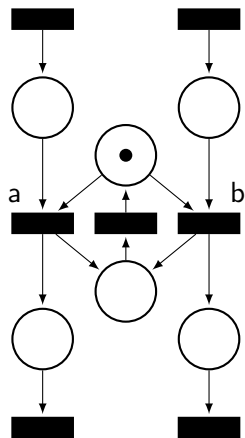
Semaphore



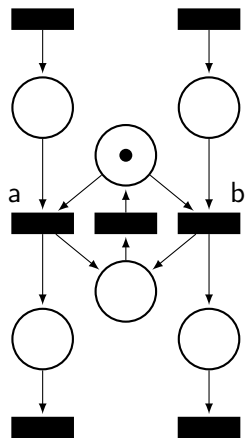
Rende-vous



Basic Constructions

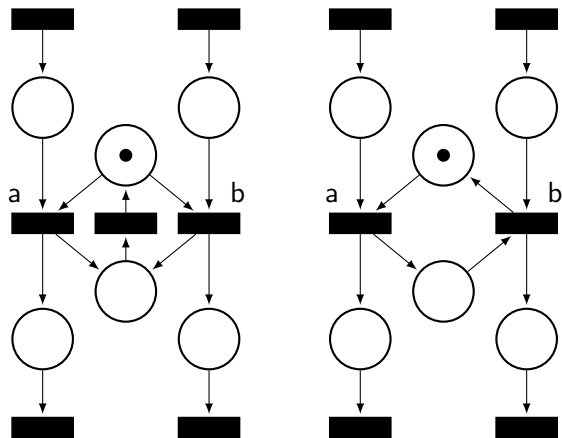


Basic Constructions



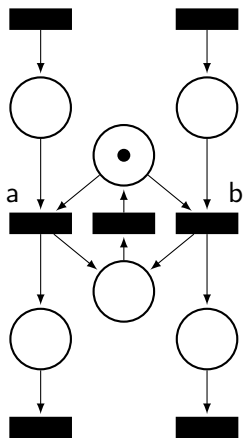
Critical section

Basic Constructions

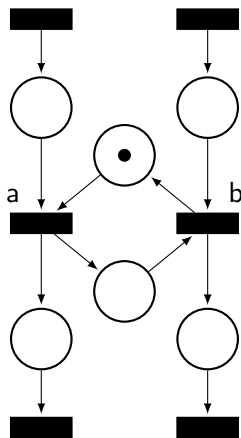


Critical section

Basic Constructions

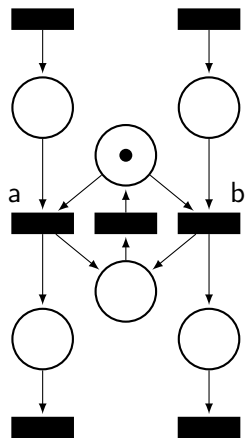


Critical section

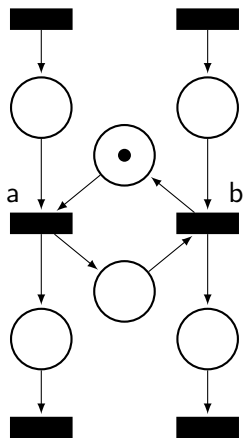


Alternation

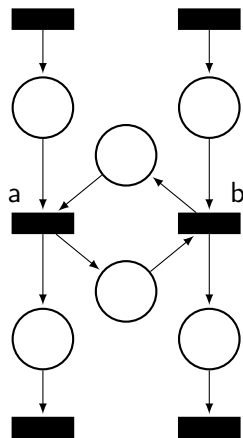
Basic Constructions



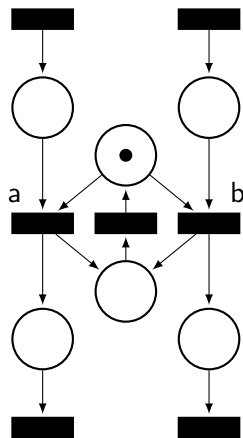
Critical section



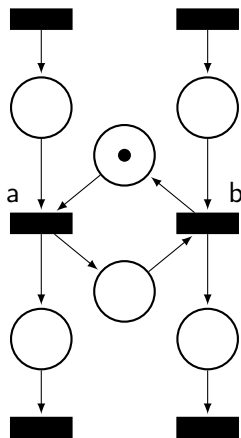
Alternation



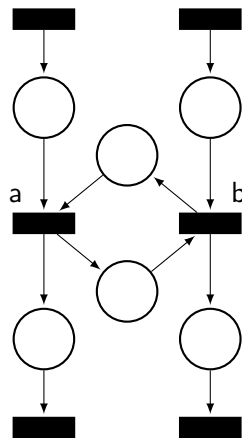
Basic Constructions



Critical section



Alternation



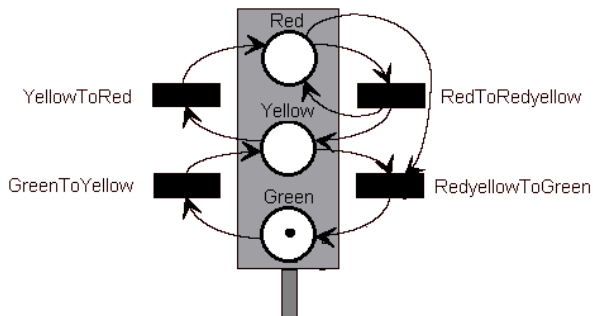
Deadlock

Different Modifications/Extensions of Petri Nets

- Condition/Event Petri nets (C/E PN)
- Place/Transition Petri nets (P/T PN)
- Coloured Petri nets
- Hierarchical Petri nets
- Timed Petri nets
- Time Petri nets
- Stochastic Petri nets

Condition/Event Petri Nets

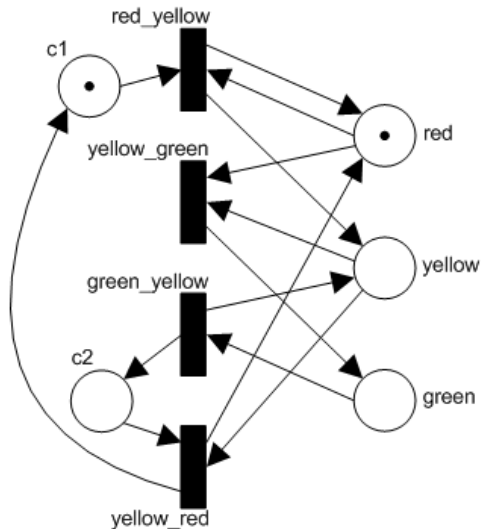
- places = conditions
- transitions = event



In each place, there is one or none token.

Condition/Event Petri Nets

Better and a bit more complicated example.

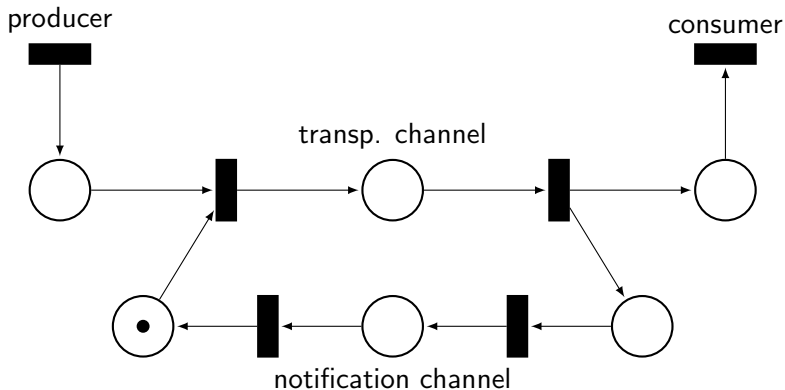


Transition/Place Petri Nets

An arbitrary number of tokens in each place.

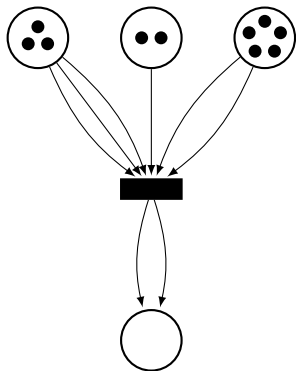
Transition/Place Petri Nets

An arbitrary number of tokens in each place.

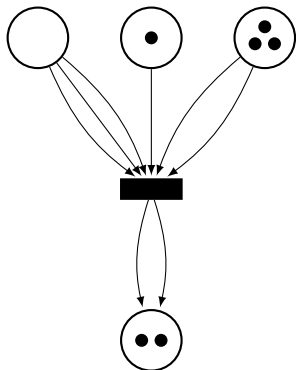


Producer-consumer model for bounded transport channel.

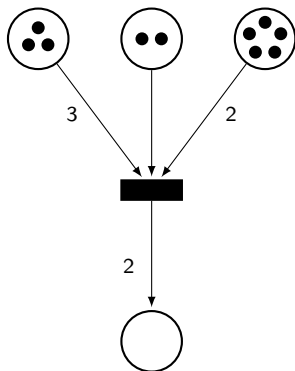
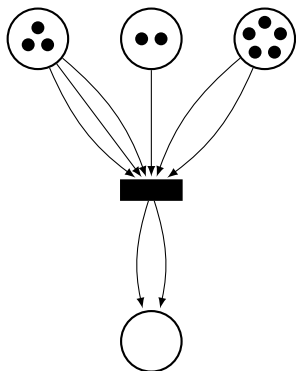
Additional Constructs - Arc Multiplicity



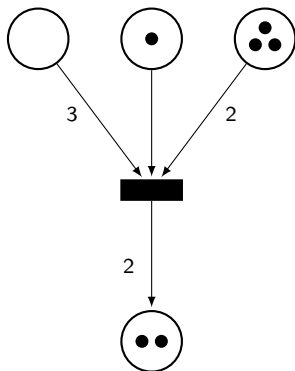
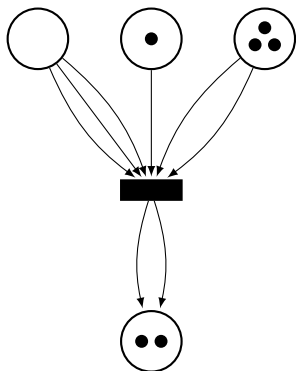
Additional Constructs - Arc Multiplicity



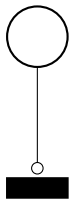
Additional Constructs - Arc Multiplicity



Additional Constructs - Arc Multiplicity

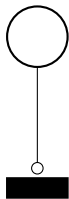


Additional Constructs - Inhibitor and Reset Arcs

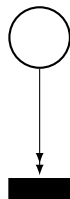


An **inhibitor arc** imposes the precondition that the transition may only fire when the place is empty.

Additional Constructs - Inhibitor and Reset Arcs



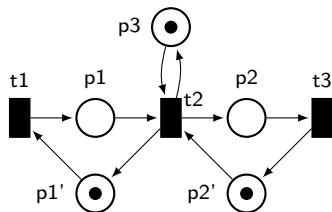
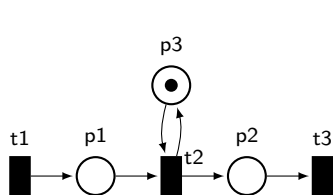
An **inhibitor arc** imposes the precondition that the transition may only fire when the place is empty.



A **reset arc** does not impose a precondition on firing, and empties the place when the transition fires.

Properties of Petri nets

- **reachability** - reachability tree or coverability tree
- **bounded (safe) places**
 - a place with a **bound** on the number of its tokens in all reachable markings
 - a place is **safe** if the number of its tokens ≤ 1 in all reachable markings
- **liveness**
 - a transition is **live** if, from every marking, one can reach a marking where the transition is enabled
 - a net is **live** if all its transitions are live



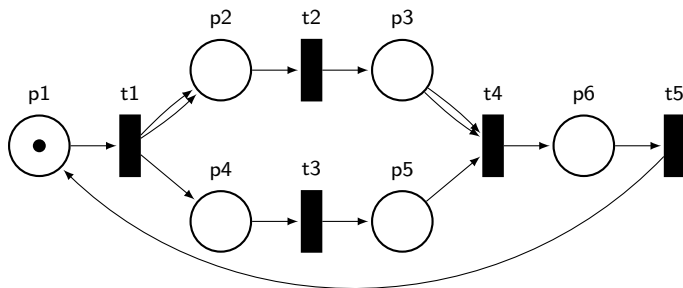
Properties of Petri nets

- **p-invariant**

- an invariant vector on places, i.e. a multiset of places representing weighting such that any such weighted marking remains invariant by any firing, e.g. $3 * p_1 + p_2 + p_3 + p_4 + p_5 + 3 * p_6$.

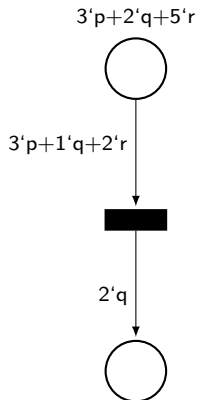
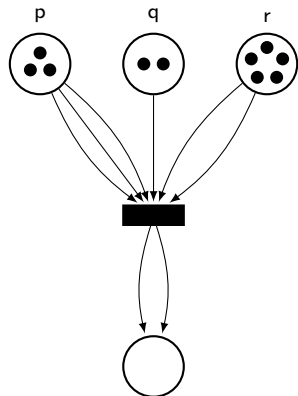
- **t-invariant**

- an invariant vector on transitions, i.e. a multiset of transitions whose firing leave invariant any marking, e.g. $t_1 + 2 * t_2 + t_3 + t_4 + t_5$.



Coloured Petri Nets

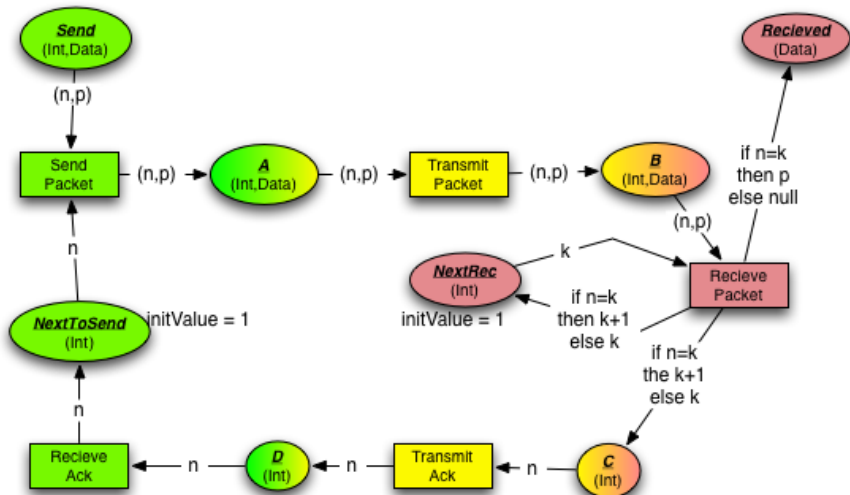
Different colours (classes) of tokens.



marking expression, arc expression, transition guard (next slide)

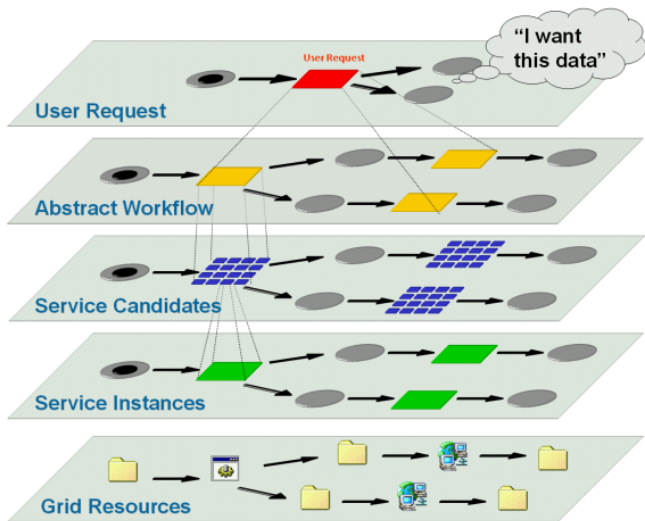
Colours usually serves for data type representation.

Coloured Petri Net Example



source: http://scienceblogs.com/goodmath/2007/10/colored_petri_nets.php

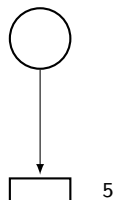
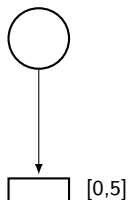
Hierarchical Petri Nets



source: <http://www.gridworkflow.org/kwfgird/gwes-web/>

Time PN, Timed PN, Stochastic PN

time from enabling of a transition to fire,
interval of the time to fire,
time of firing,
probability distribution on time to fire,
exponential distribution on time to fire,
...



PN Tools

CPN Tools

- Coloured Petri Nets (with prioritized transitions and real time support)
- editor, simulation, analyses

TimeNET

- High-level PN, Stochastic PN, PN with Time
- editor, simulation, analyses (p-invariant, performance analyses)

Tapall

- Timed-Arc Petri Nets (with real time support)
- editor, simulation, CTL logic checker

And many more tools and use cases, see, e.g.

<http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html>
<http://cs.au.dk/cpnets/industrial-use/>

Relevant Lectures

IV113 Úvod do validace a verifikace (Barnat)

IV109 Modelování a simulace (Pelánek)

IA159 Formal verification methods (Strejček)

PV177 Laboratoř pokročilých síťových technologií (Řehák)

References

ITU-T specifications:

- Z.100 Specification Description Language (SDL). 2007.
- Z.120 Message Sequence Charts (MSC). 2004.

Books:

- S. Mullender. *Distributed Systems*. ACM, 1993.
- D. Peled. *Software Reliability Methods*. Springer, 2001.
- R. Bræk et al. *TIME: The Integrated Method*. SINTEF, 1999.
- L. Doldi. *Validation of Communications Systems with SDL*. Wiley, 2003.
- M. Broy and K. Stølen. *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer, 2001.
- W. Jia and W. Zhou. *Distributed Network Systems: From Concepts to Implementation*. Springer, 2005.
- M. Češka. *Petriho sítě*. Akademické nakladatelství CERM Brno, 1994.
- J. Markl. *Petriho sítě*. VŠB - Technická univerzita Ostrava, 1996.