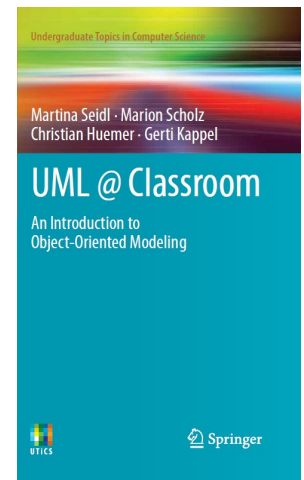

Activity diagrams in UML

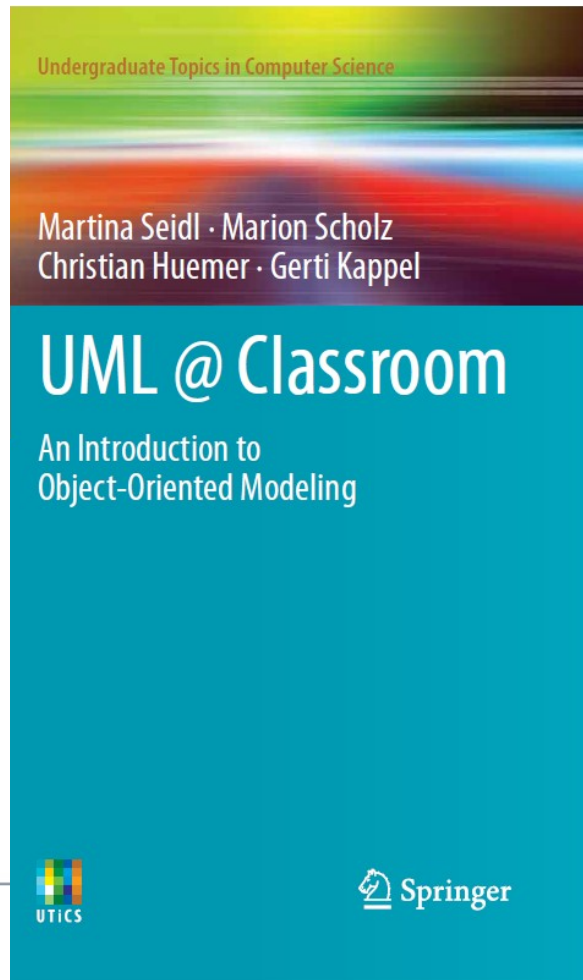
Slides accompanying UML@Classroom
Version 1.0.1

Originally designed by the Business Informatics Group @ TU Wien
Altered by Cor-Paul Bezemer



Literature

- The slides are based on the following book:



UML @ Classroom: An Introduction to Object-Oriented Modeling

Martina Seidl, Marion Scholz, Christian
Huemer and Gerti Kappel

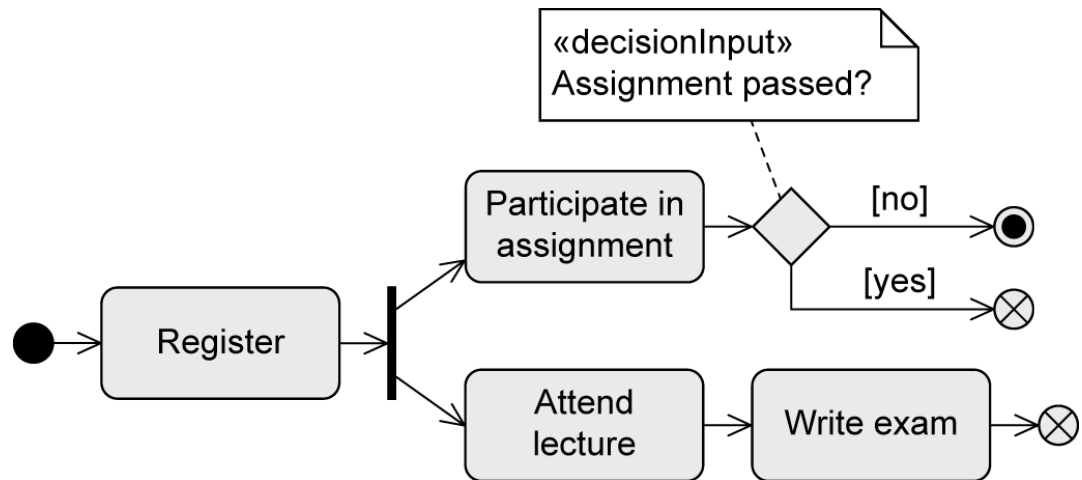
Springer Publishing, 2015

ISBN 3319127411

The book is available as an eBook in the
U of A library

<https://www.library.ualberta.ca/>

An example activity diagram

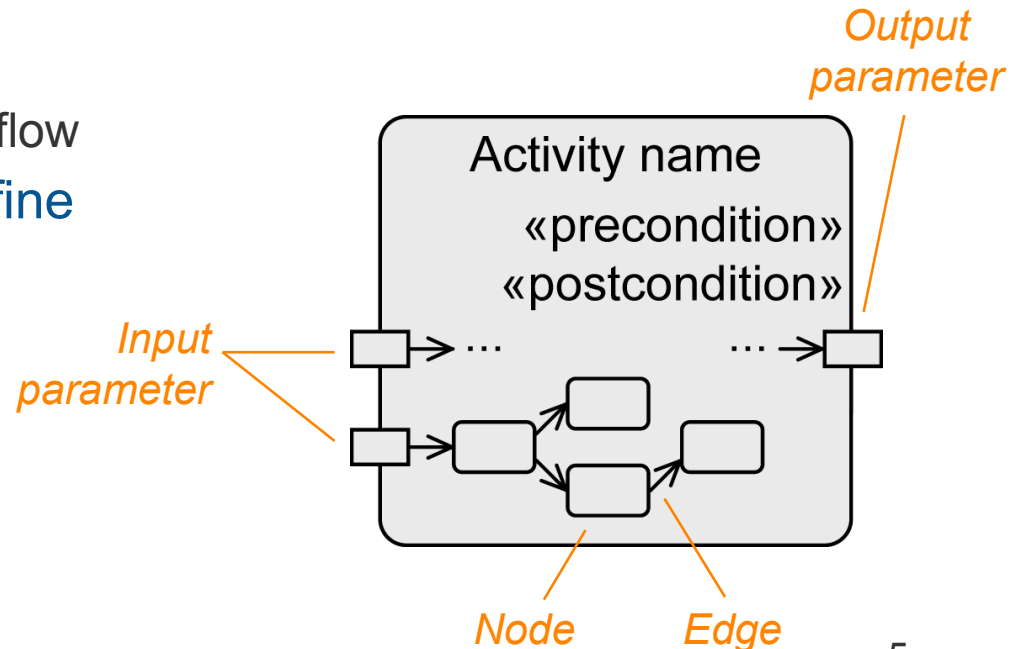


Introduction

- Focus of activity diagram: **representing workflows**
- Flow-oriented language concepts
- Based on
 - languages for defining business processes
 - established concepts for describing concurrent communicating processes (token concept as found in petri nets)
- Concepts and notation variants cover **broad area of applications**
 - Modeling of object-oriented and non-object-oriented systems

Activity

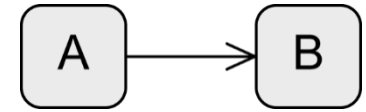
- Specification of user-defined behavior at different levels of granularity
- Examples:
 - Definition of the behavior of an operation in the form of individual instructions
 - Modeling the course of actions of a use case
 - Modeling the functions of a business process
- An activity is a directed graph
 - Nodes: actions and activities
 - Edges: for control and object flow
- Control flow and object flow define the execution
- Optional:
 - Input/output parameters
 - Pre- and postconditions



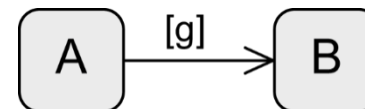
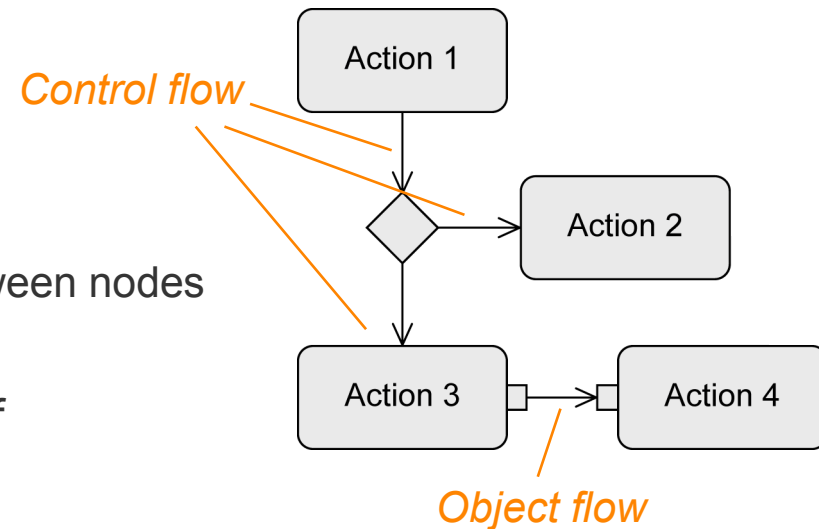
Action

- **Basic element** to specify user-defined behavior
- **Atomic** but can be aborted
- No specific rules for the description of an action
 - Definition in natural language or in any programming language
- Process input values to produce output values
- Special notation for predefined types of actions, most importantly
 - Event-based actions
 - Call behavior actions

Edges

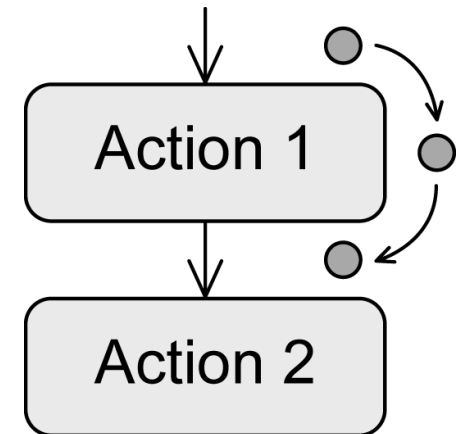


- Connect activities and actions to one another
- Express the execution order
- Types
 - Control flow edges
 - Define the order between nodes
 - Object flow edges
 - Used to exchange data or objects
 - Express a data/causal dependency between nodes
- Guard (condition)
 - Control and object flow only continue if guards in square brackets evaluate to true



Token

- **Virtual coordination mechanism** that describes the execution exactly
 - No physical component of the diagram
 - Mechanism that grants the execution permission to actions
- If an action receives a token on all its inputs, the action can be executed
- When the action has completed, it passes the token to a subsequent action and the execution of this action is triggered
- Guards can prevent the passing of a token
 - Tokens are stored in previous node
- Control token and object token
 - **Control token:** “execution permission” for a node
 - **Object token:** transport data + “execution permission”



Beginning and Termination of Activities

● Initial node

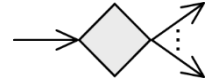
- Starts the execution of an activity
- Provides tokens at all outgoing edges
- Keeps tokens until the successive nodes accept them
- Multiple initial nodes to model concurrency

◎ Activity final node

- Ends all flows of an activity
- First token that reaches the activity final node terminates the entire activity
 - Concurrent subpaths included
- Other control and object tokens are deleted
 - Exception: object tokens that are already present at the output parameters of the activity

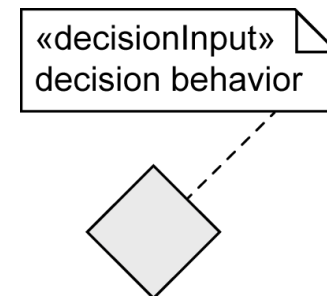
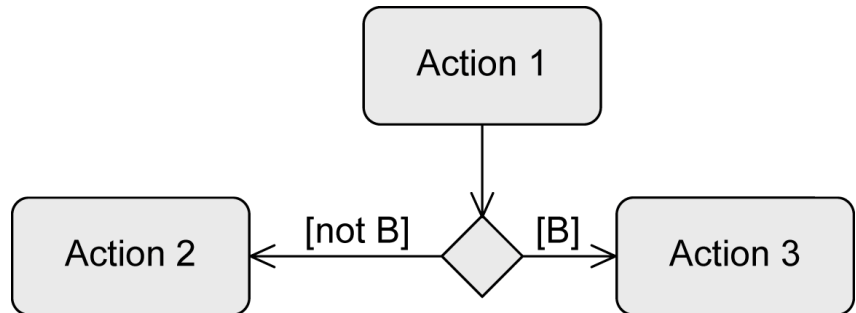
⊗ Flow final node

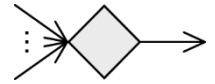
- Ends one execution path of an activity
- All other tokens of the activity remain unaffected



Alternative Paths – Decision Node

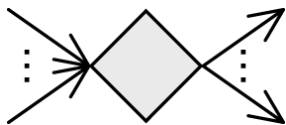
- To define alternative branches
- „Switch point“ for tokens
- Outgoing edges have guards
 - Syntax: [Boolean expression]
 - Token takes **one** branch
 - Guards must be mutually exclusive
 - Predefined: [else]
- Decision behavior
 - Specify behavior that is necessary for the evaluation of the guards
 - Execution must not have side effects
 - Introduced to reduce repetition in guards



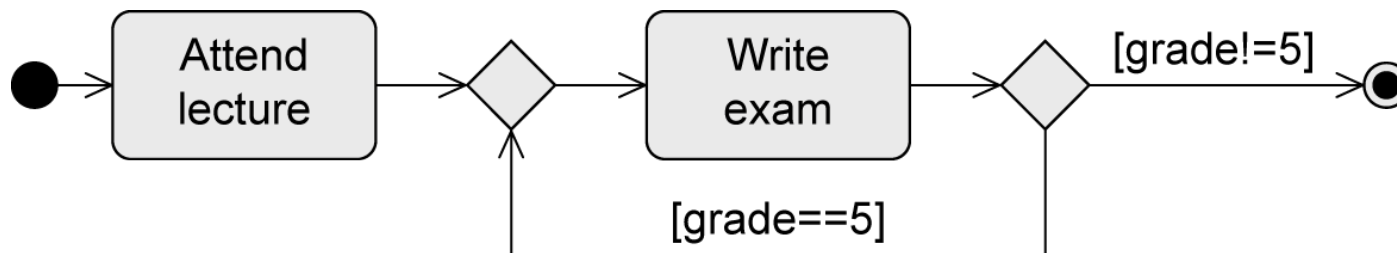


Alternative Paths – Merge Node

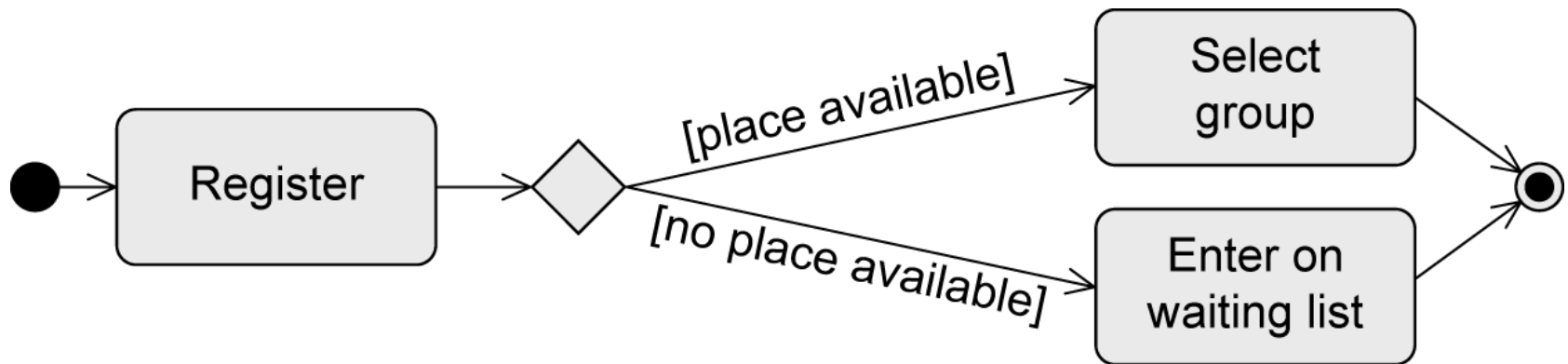
- To bring **alternative** subpaths together
- Passes token to the next node
- Combined decision and merge node

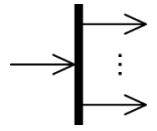


- Decision and merge nodes can also be used to model loops:



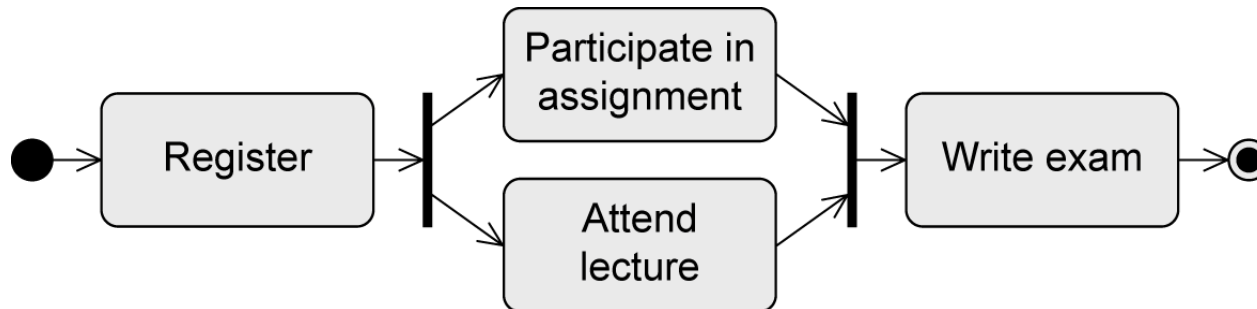
Example: Alternative Paths

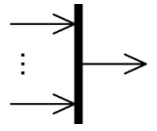




Concurrent Paths – Parallelization Node

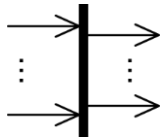
- To split path into concurrent subpaths
- Duplicates token for all outgoing edges
- Example:



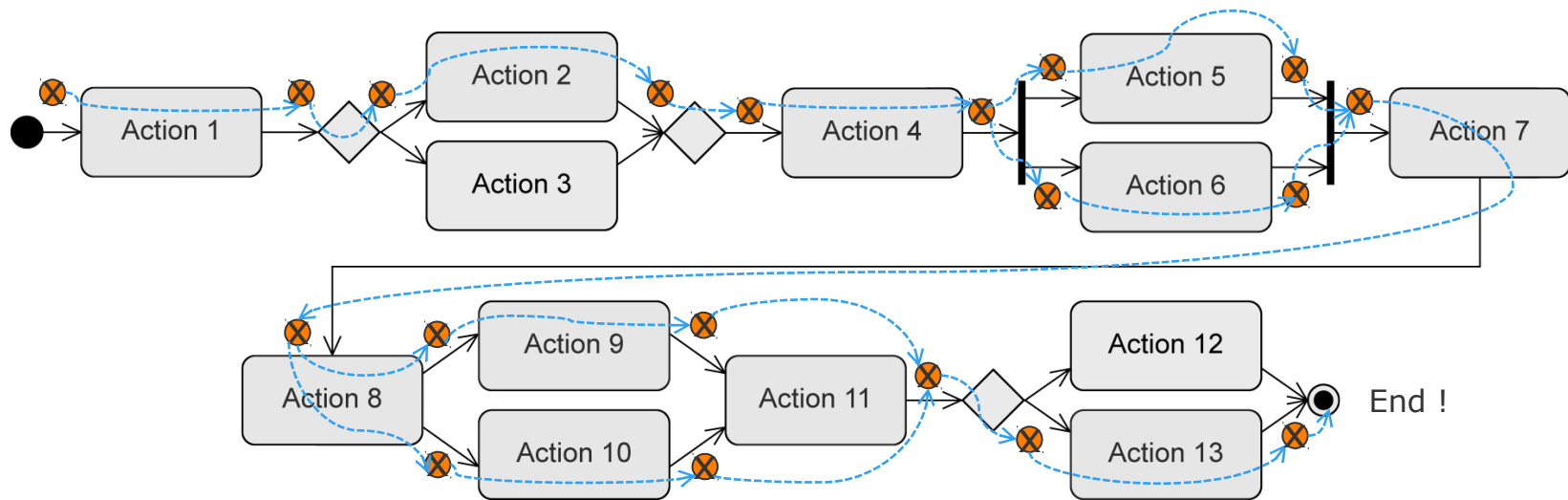


Concurrent Paths – Synchronization Node

- To merge concurrent subpaths
- Token processing
 - Waits until tokens are present at all incoming edges
 - Merges all control tokens into one token and passes it on
 - Passes on all object tokens
- Combined parallelization and synchronization node:

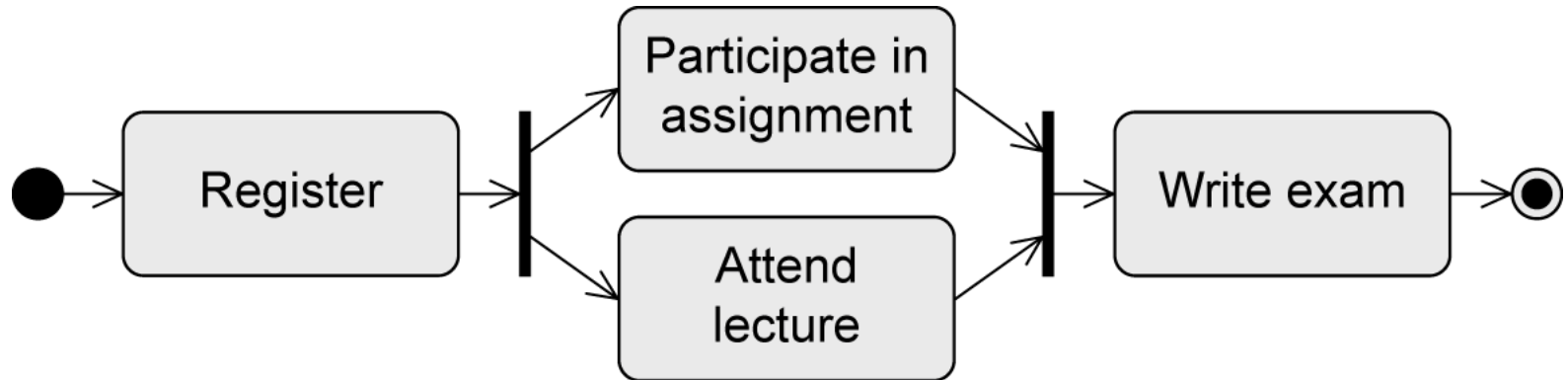


Example: Token (Control Flow)

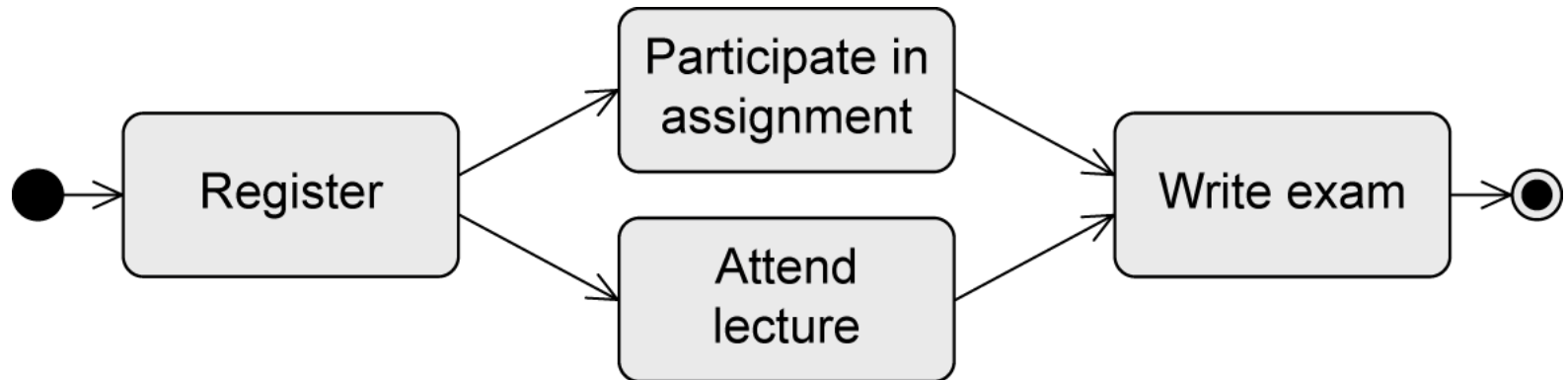


- ... all outgoing edges of all initial nodes are assigned a token....
- ... if all incoming edges of an action have a token, the action is activated and is ready for execution
- ... before the execution, the action consumes one token from every incoming edge;
after the execution, the action passes one token to every outgoing edge
- ... a decision node passes the token to **one** outgoing edge (depending on the result of the evaluation of the guard)
- ... a merge node individually passes each token it gets to its outgoing edge
- ... a parallelization node duplicates an incoming token for **all** outgoing edges
- ... a synchronization node waits until all incoming edges have a token, merges them to a single token and passes it to its outgoing edge
- ... the first token that reaches the activity final node terminates the entire activity

Example: Equivalent Control Flow?

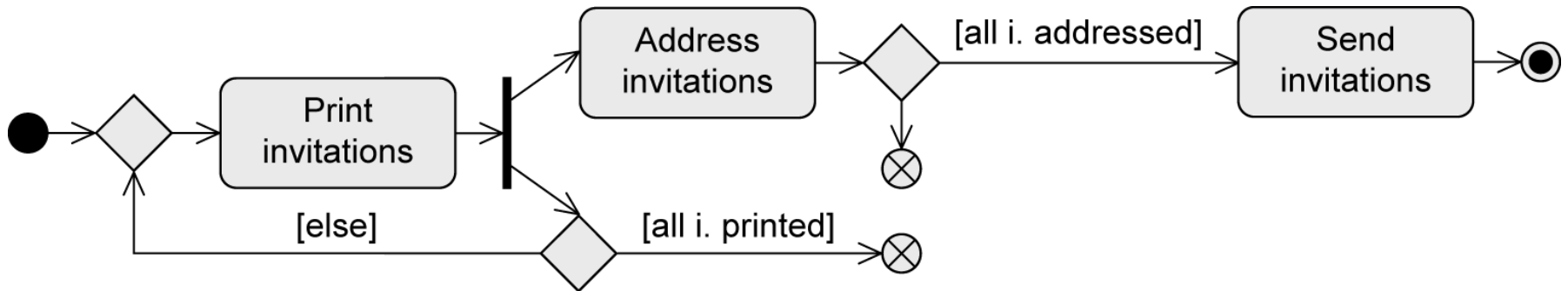


... equivalent to ...

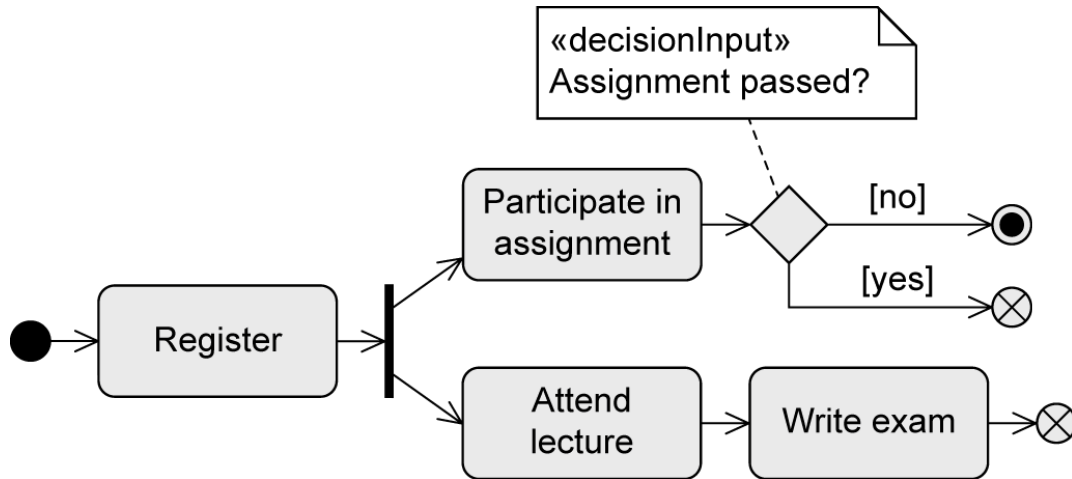


Example: Create and Send Invitations to a Meeting

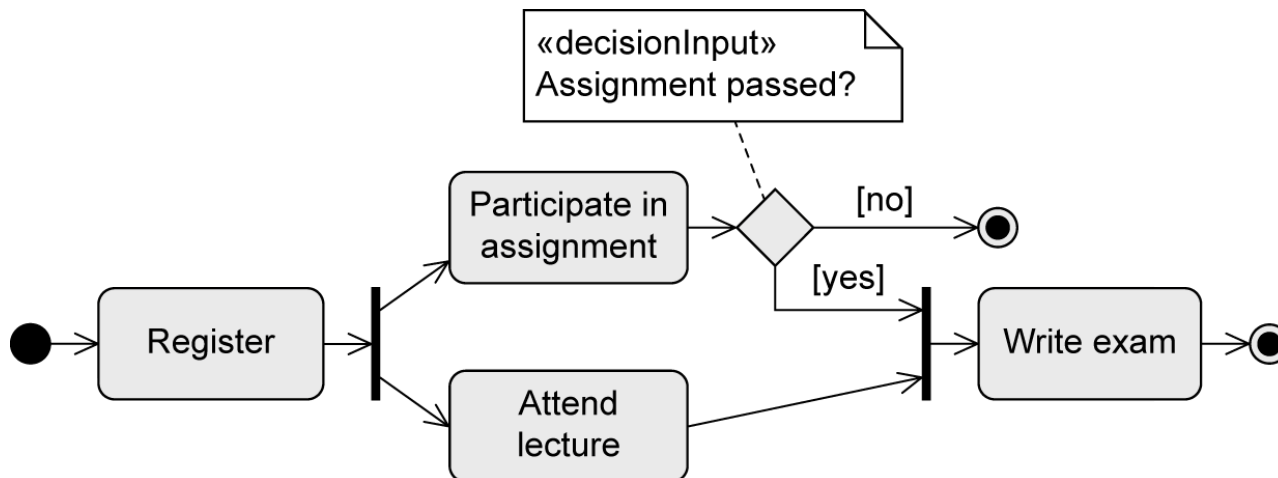
- While invitations are printed, already printed invitations are addressed.
- When all invitations are addressed, then the invitations are sent.



Example: Conduct Lecture (Student Perspective)



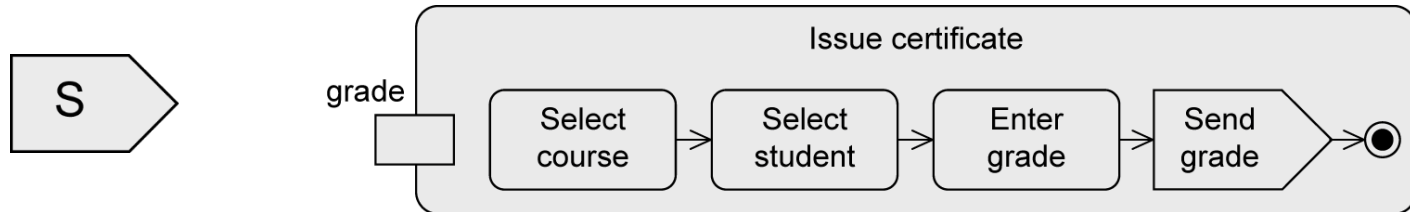
NOT equivalent ... why?



Event-Based Actions

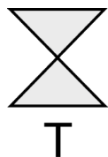
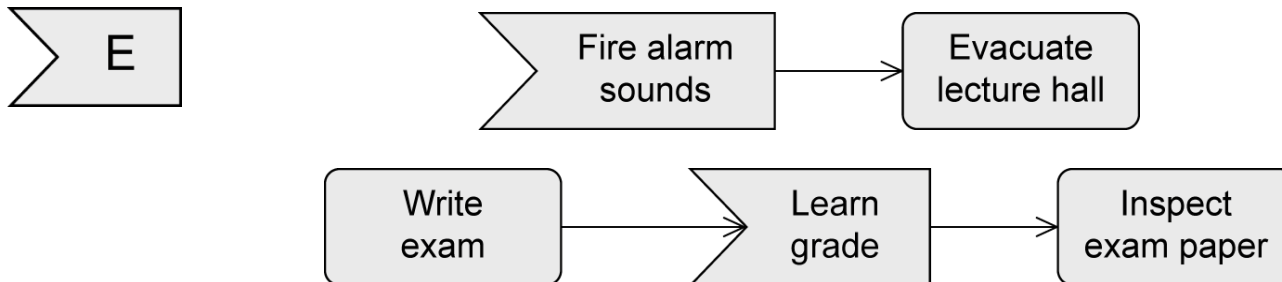
- To send signals

- Send signal action

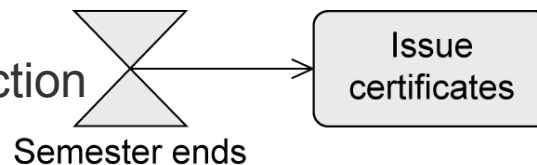


- To accept events

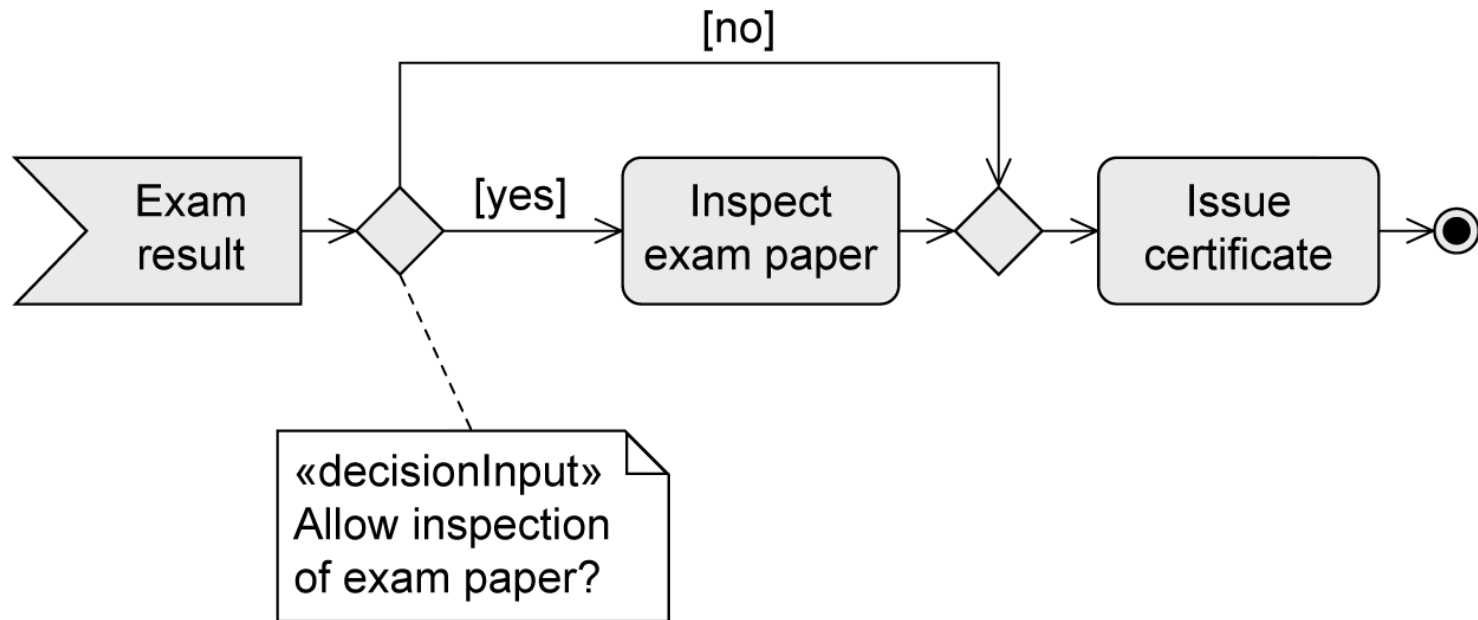
- Accept event action



- Accept time event action

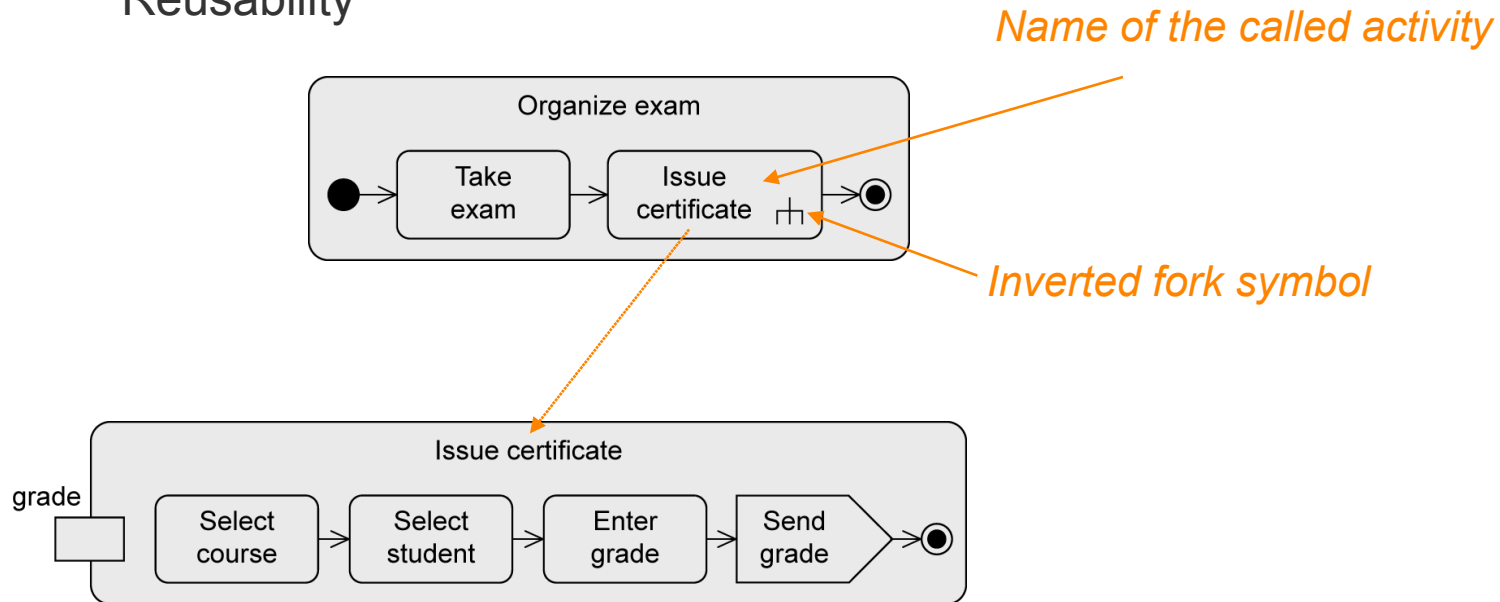


Example: Accept Event Action

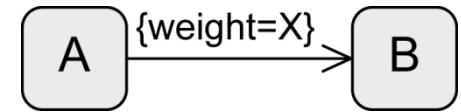


Call Behavior Action

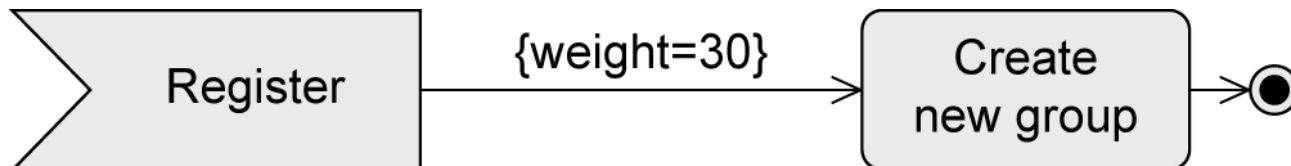
- The execution of an action can call an activity
- Content of the called activity can be modeled elsewhere
- Advantages:
 - Model becomes clearer
 - Reusability



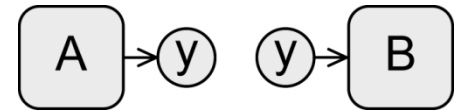
Weight of Edges



- Minimal number of tokens that must be present for an action to be executed
- Default: 1
- All tokens present have to be consumed: 0 (also **all** or *****)



Connector

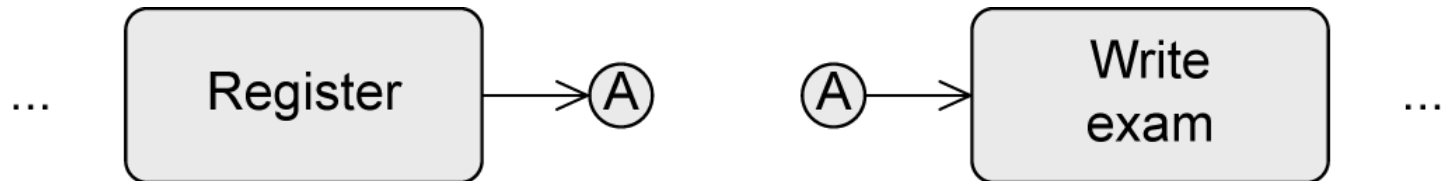


- Used if two consecutive actions are far apart in the diagram

- Without connector:



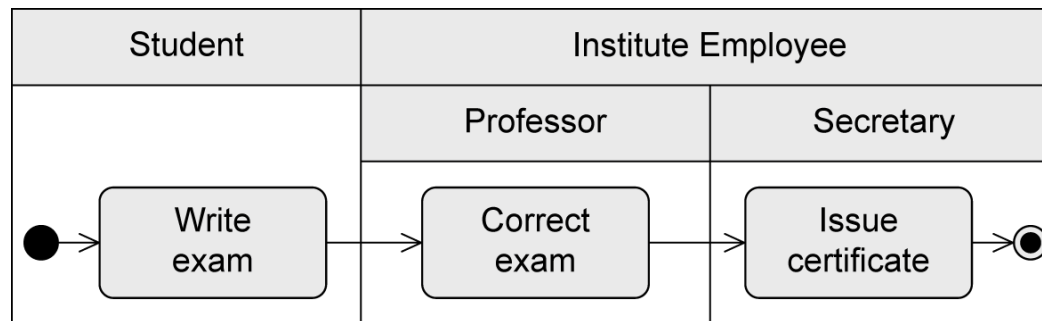
- With connector



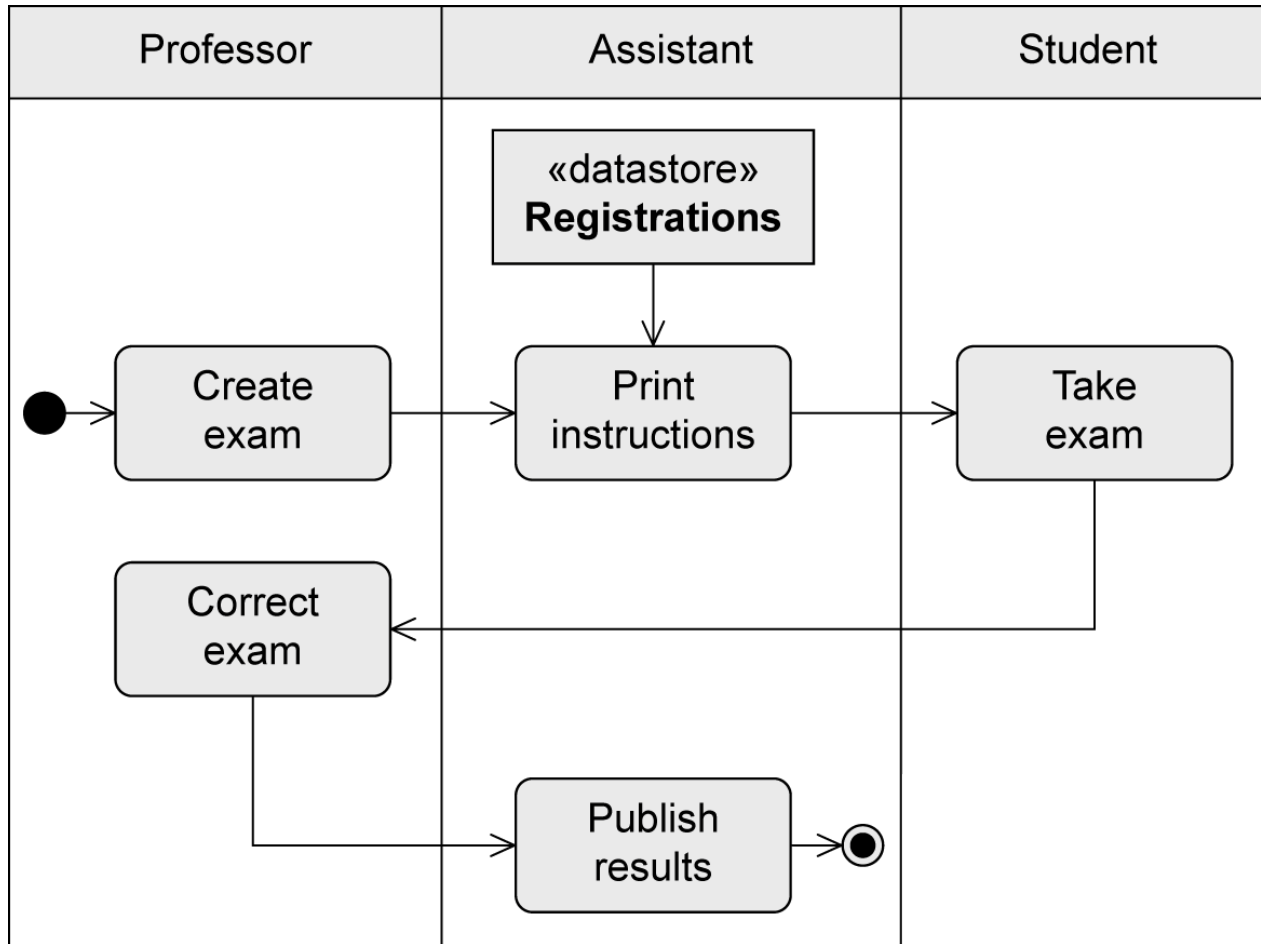
A	B	A
		B

Partition

- Allows the grouping of nodes and edges of an activity due to responsibilities
- Responsibilities reflect organizational units or roles
- Makes the diagram more structured
- Does not change the execution semantics
- Example: partitions **Student** and **Institute Employee** (with subpartitions **Professor** and **Secretary**)

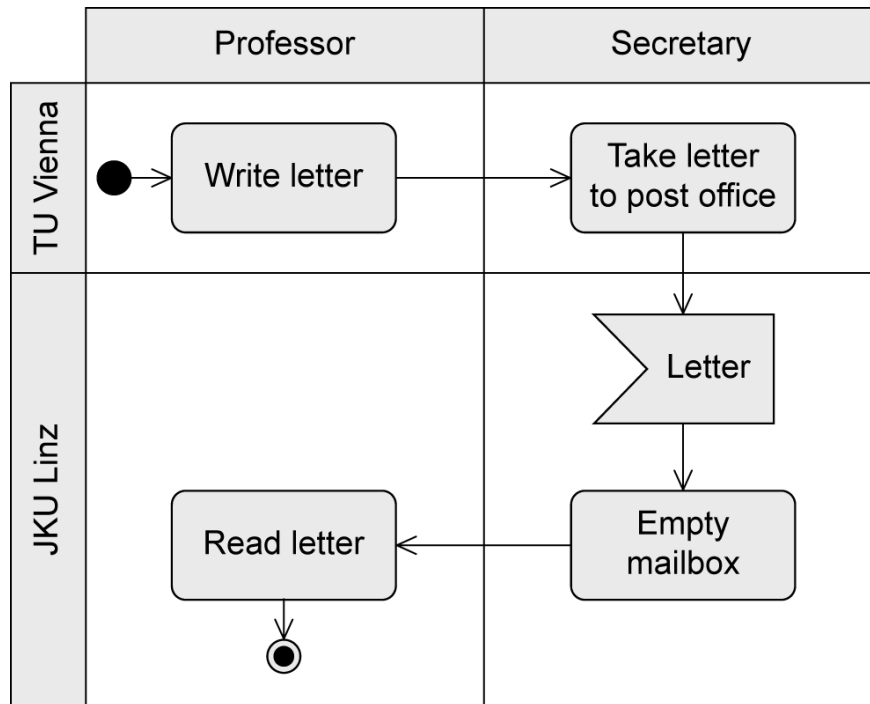


Example: Partitions



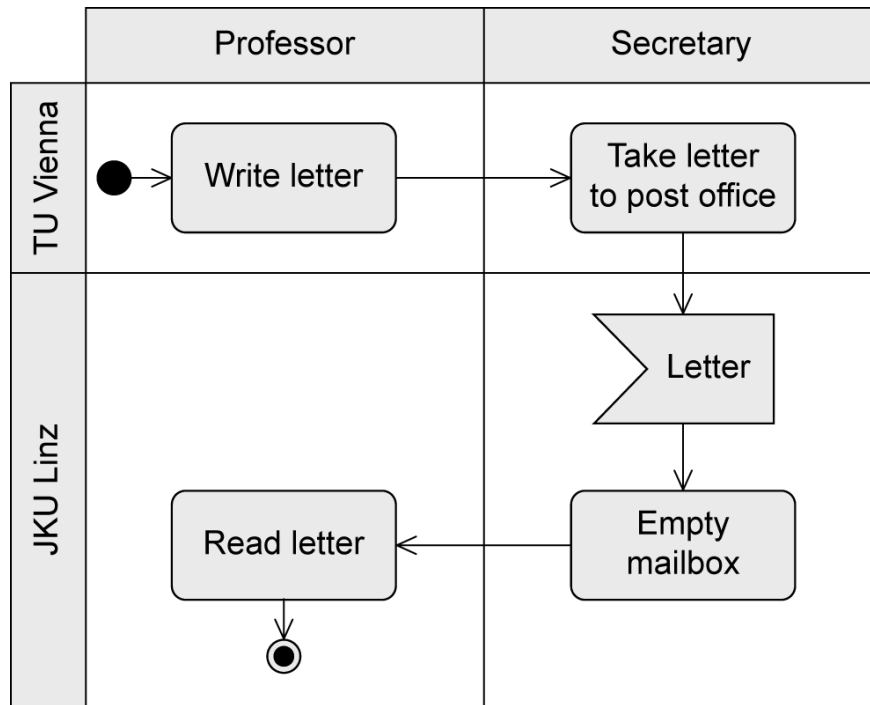
Multidimensional Partitions

- Graphical notation

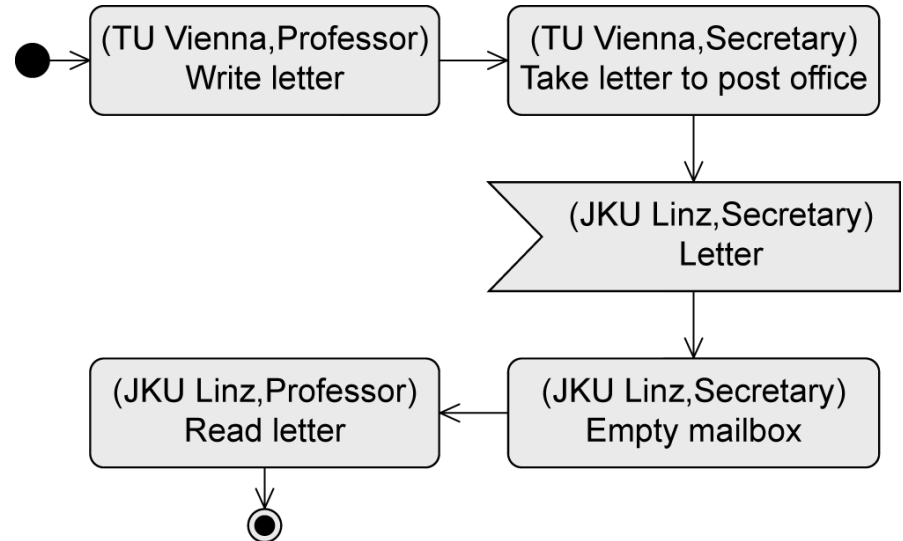


Multidimensional Partitions

■ Graphical notation

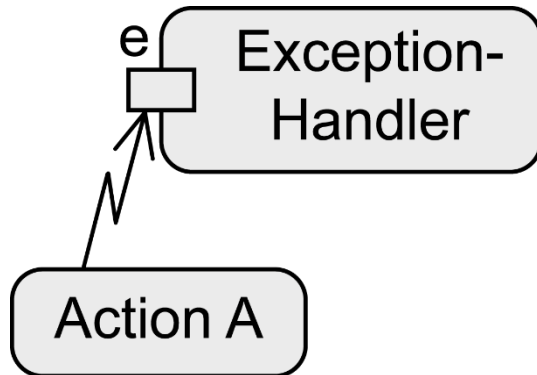


... or alternatively textual notation



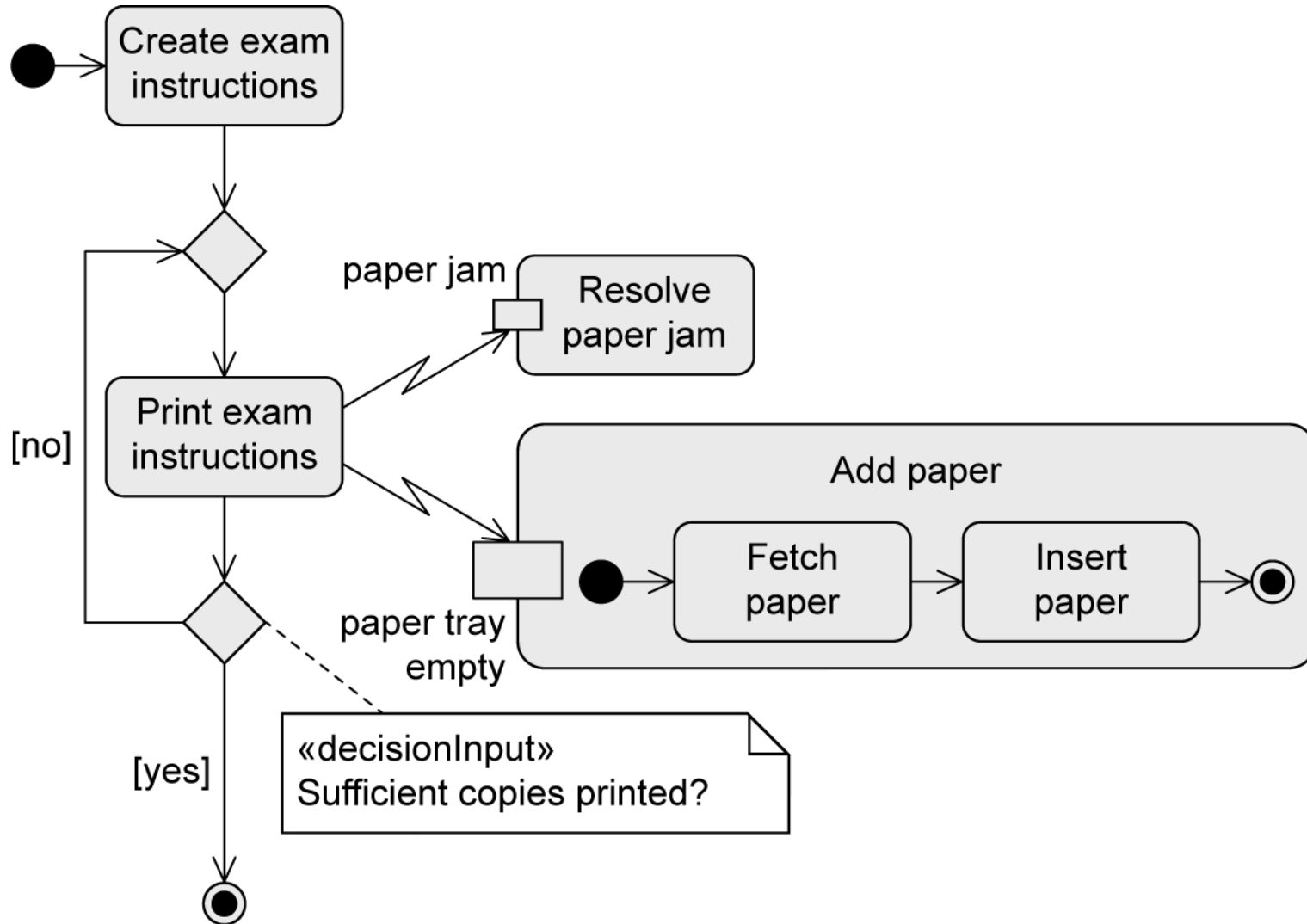
Exception Handling – Exception Handler

- Predefined exceptions
- Defining how the system has to react in a specific error situation
- The exception handler replaces the action where the error occurred



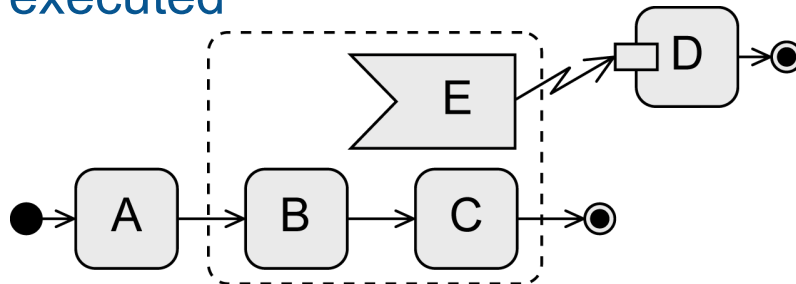
- If the error **e** occurs...
 - All tokens in **Action A** are deleted
 - The exception handler is activated
 - The exception handler is executed instead of **Action A**
 - Execution then continues regularly

Example: Exception Handler



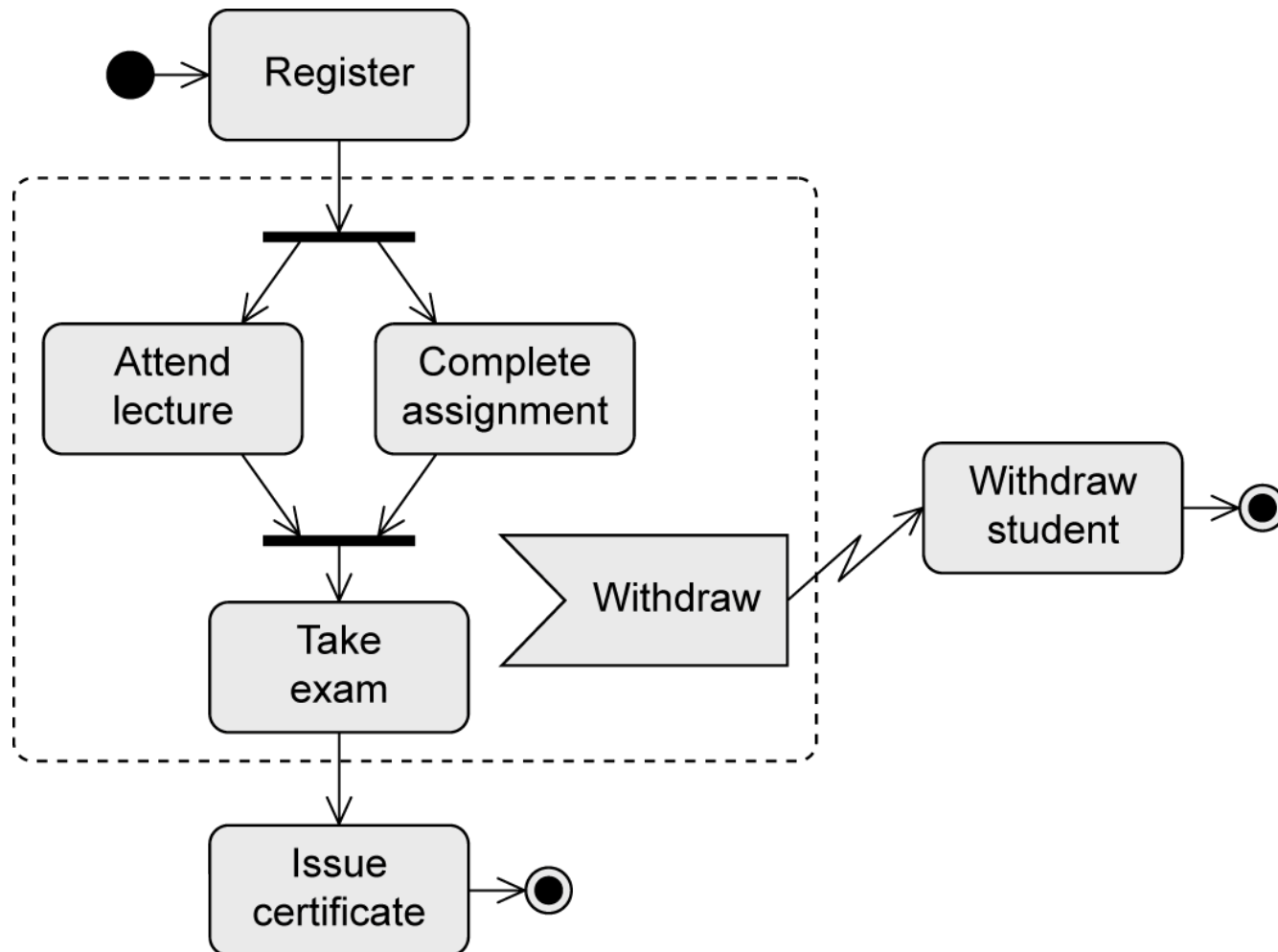
Exception Handling– Interruptible Activity Region

- Defining a group of actions whose execution is to be terminated immediately if a specific event occurs. In that case, some other behavior is executed







- If **E** occurs while **B** or **C** are executed
 - Exception handling is activated
 - All control tokens within the dashed rectangle (= within **B** and **C**) are deleted
 - **D** is activated and executed
 - No “jumping back” to the regular execution!
-

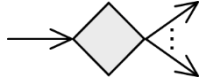
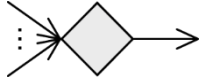
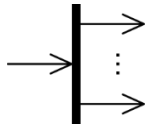
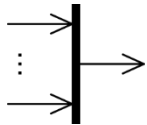
Example: Interruptible Activity Region




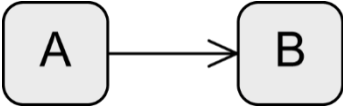
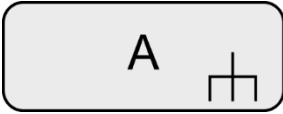
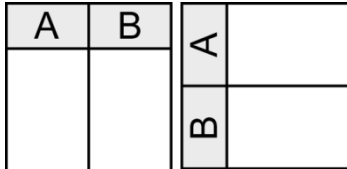
Notation Elements (1/5)

Name	Notation	Description
Action node		Represents an action (atomic!)
Activity node		Represents an activity (can be broken down further)
Initial node		Start of the execution of an activity
Activity final node		End of ALL execution paths of an activity


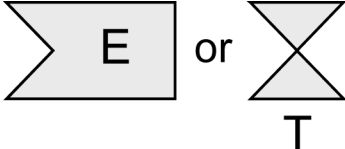
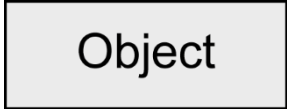


Notation Elements (2/5)

Name	Notation	Description
Decision node		Splitting of one execution path into alternative execution paths
Merge node		Merging of alternative execution paths into one execution path
Parallelization node		Splitting of one execution path into concurrent execution paths
Synchronization node		Merging of concurrent execution paths into one execution path

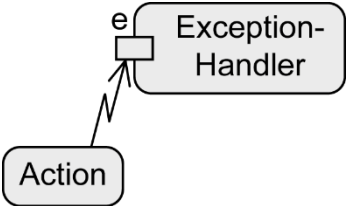
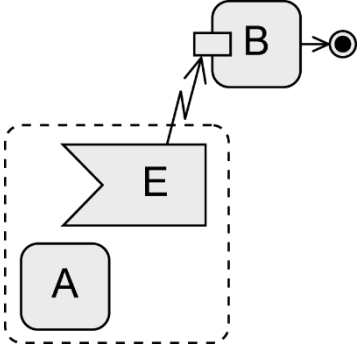
Notation Elements (3/5)

Name	Notation	Description
Flow final node		End of ONE execution path of an activity
Edge		Connection between the nodes of an activity
Call behavior action		Action A refers to an activity of the same name
Partition		Grouping of nodes and edges within an activity

Notation Elements (4/5)

Name	Notation	Description
Send signal action		Transmission of a signal to a receiver
Asynchronous accept (timing) event action		Wait for an event E or a time event T
Object node		Contains data or objects
Parameter for activities		Contains data and objects as input and output parameters
Parameter for actions (pins)		

Notation Elements (5/5)

Name	Notation	Description
Exception Handler		Exception handler is executed instead of the action in the event of an error e
Interruptible activity region		Flow continues on a different path if event E is detected