UNIVERSITÄT KOBLENZ · LANDAU

IWVI

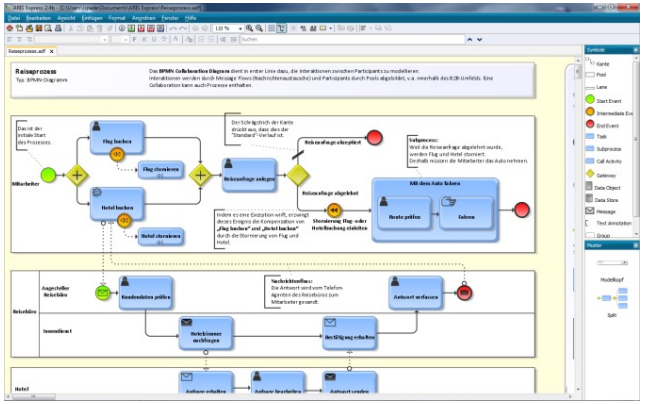
BUSINESS PROCESS MANAGEMENT

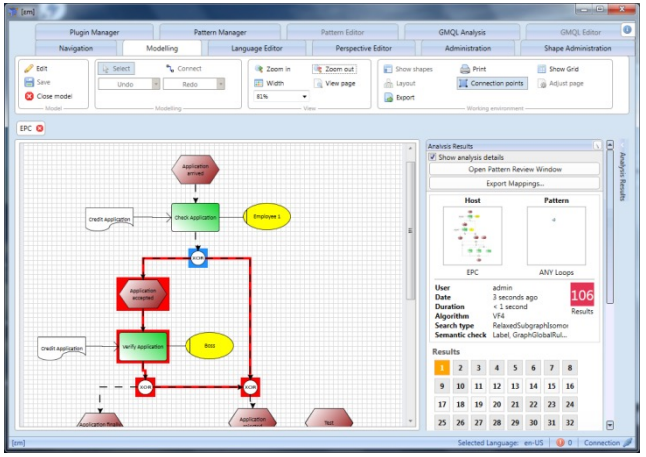
FORMALIZATION OF (PROCESS) MODELS AND (PROCESS) MODELING TOOLS

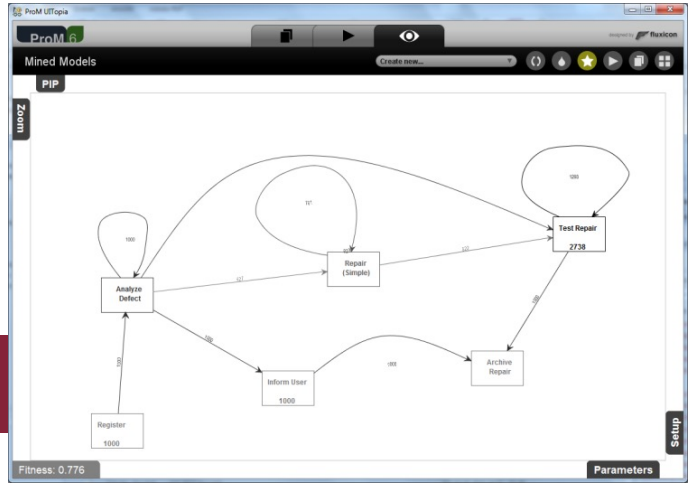
WHAT'S NEXT?

- Unambiguous models
- Model query with graph search
- Model query with temporal logic
- Business rules
- Model merge

- All this has to be done somehow automatically (e.g., with algorithms)
- Models need to be represented formally



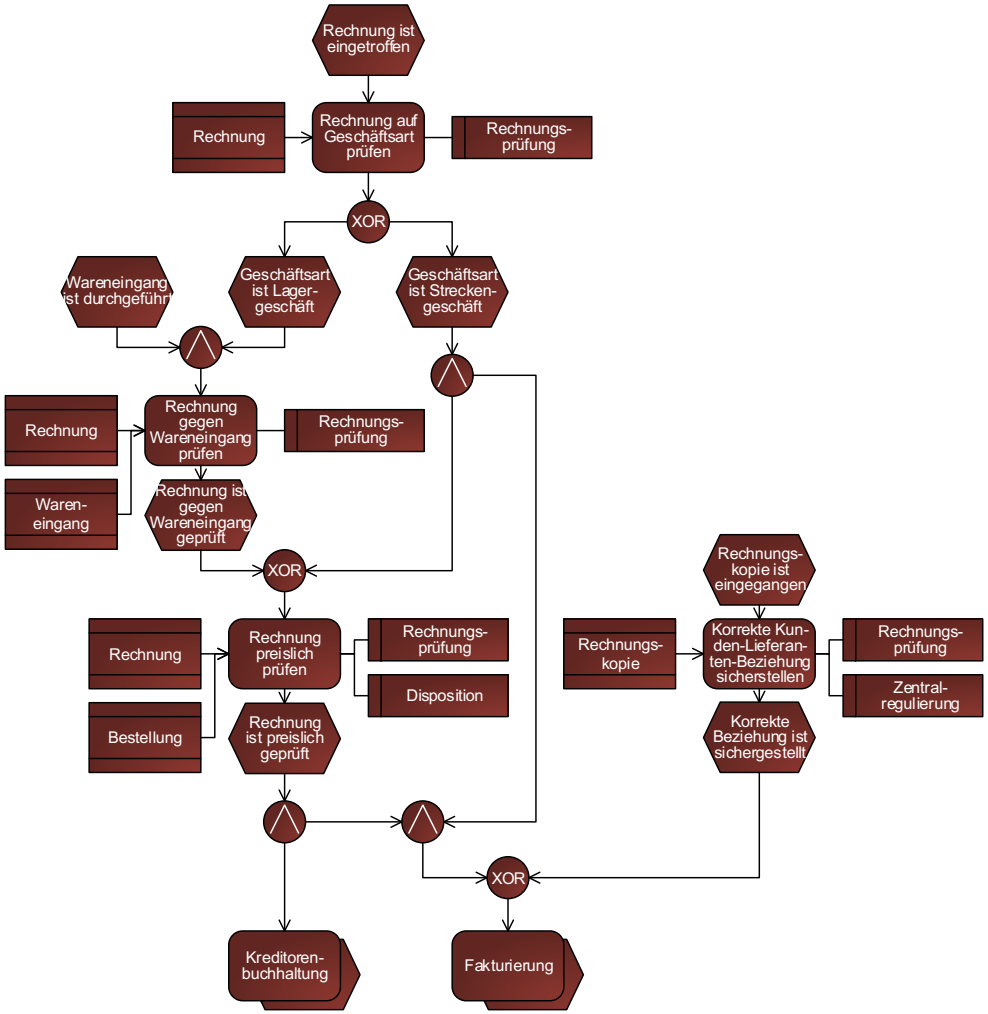




AGENDA

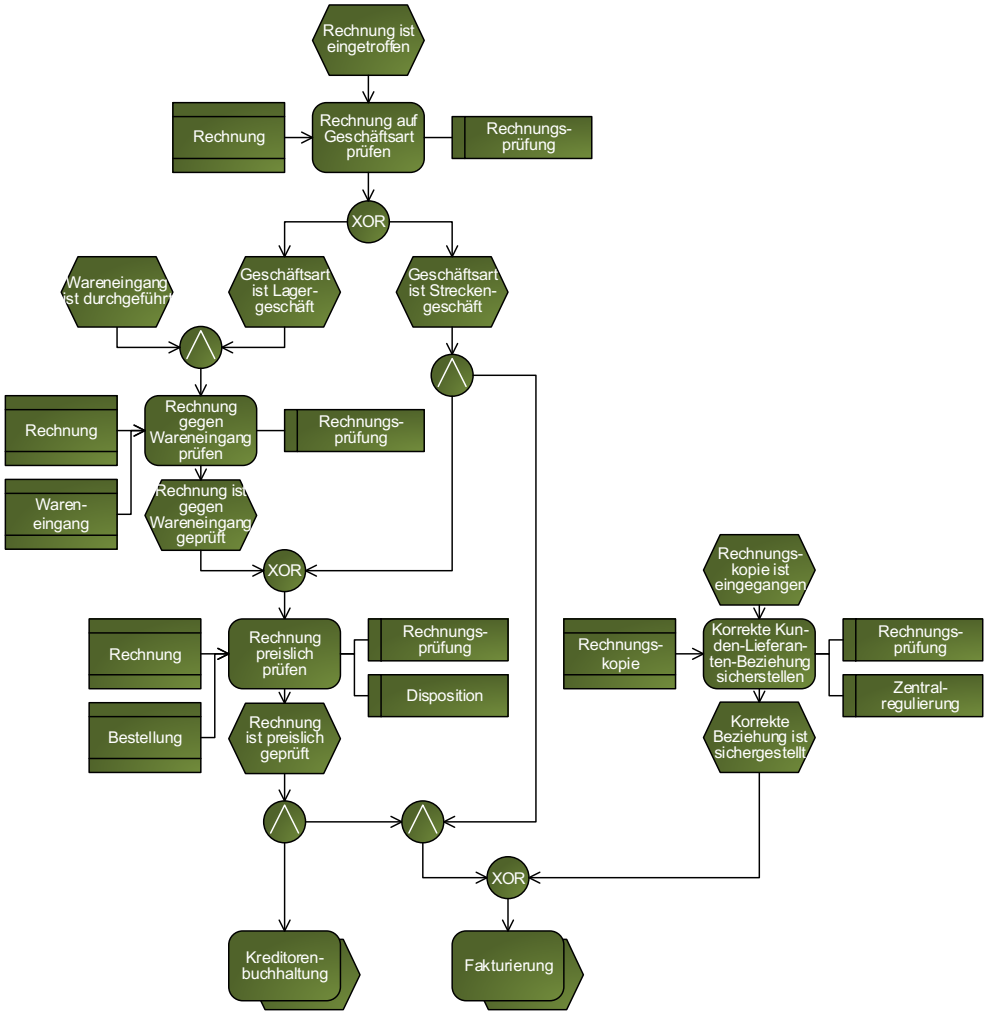
- Formalization of (Process) Models
- Storing (Process) Models

FORMAL REPRESENTATION OF (PROCESS) MODELS



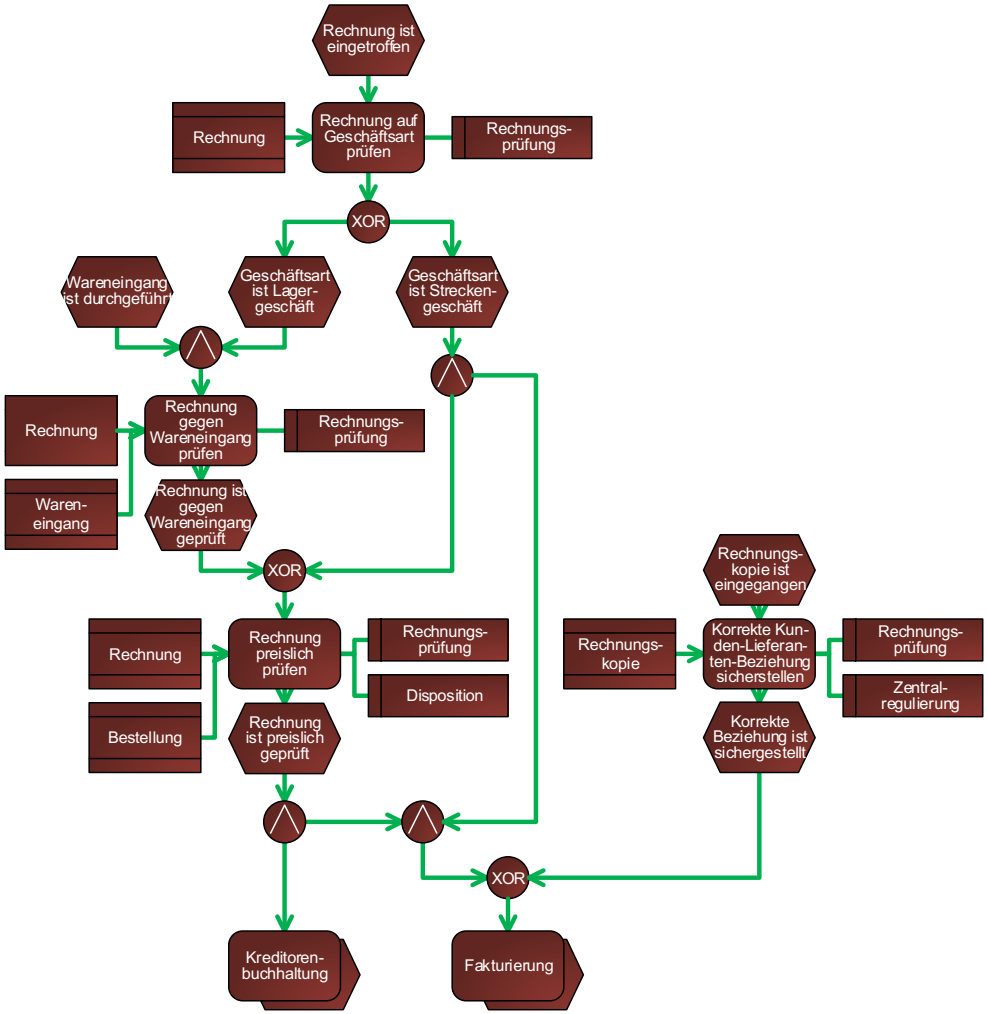
FORMAL REPRESENTATION OF (PROCESS) MODELS

- Vertices



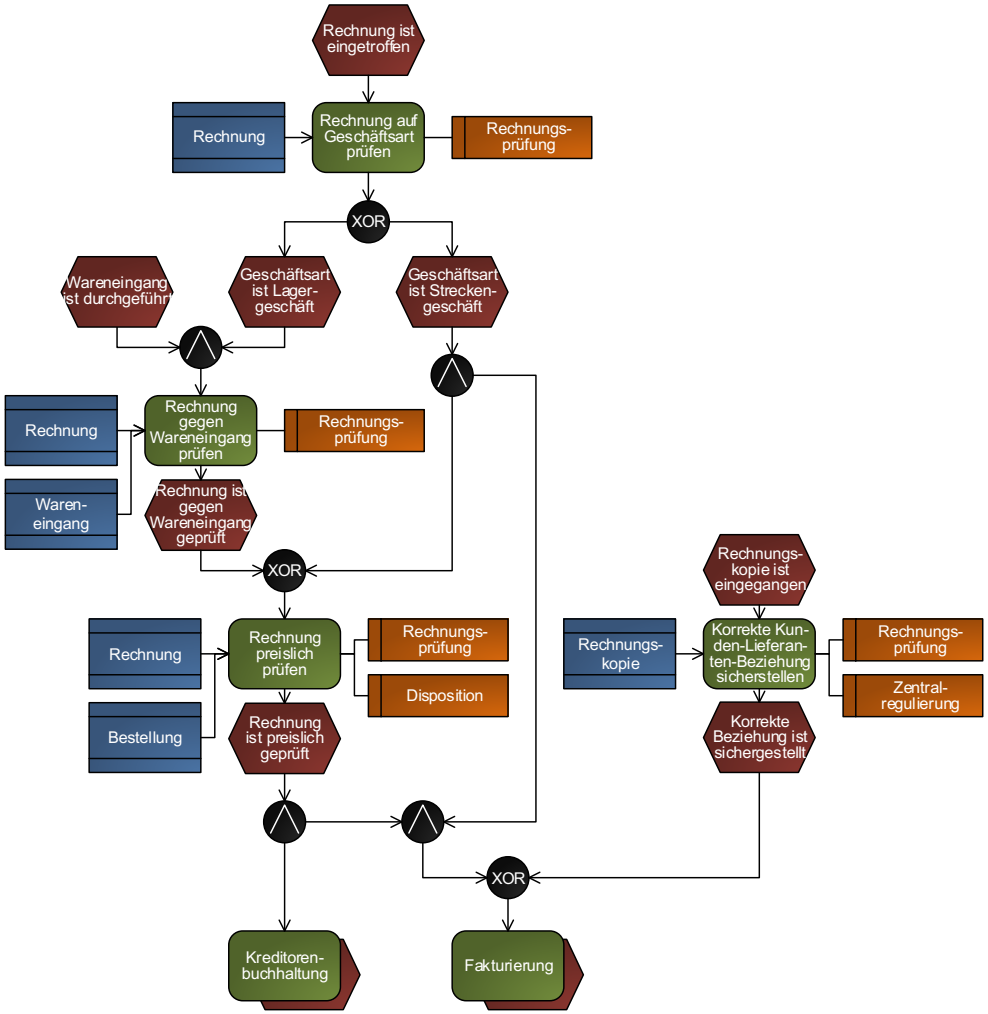
FORMAL REPRESENTATION OF (PROCESS) MODELS

- Vertices
- **Edges**



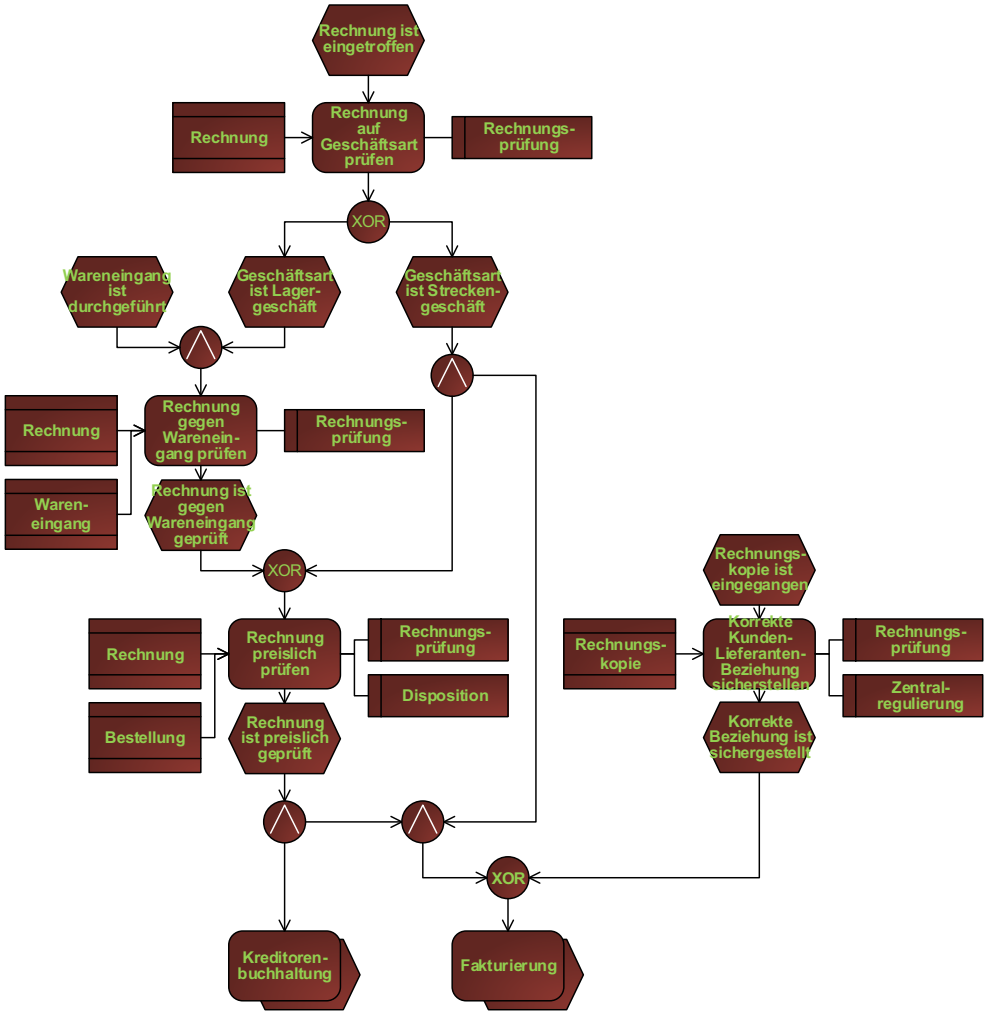
FORMAL REPRESENTATION OF (PROCESS) MODELS

- Vertices
- Edges
- **Properties of Vertices**
  - **Vertex Type**



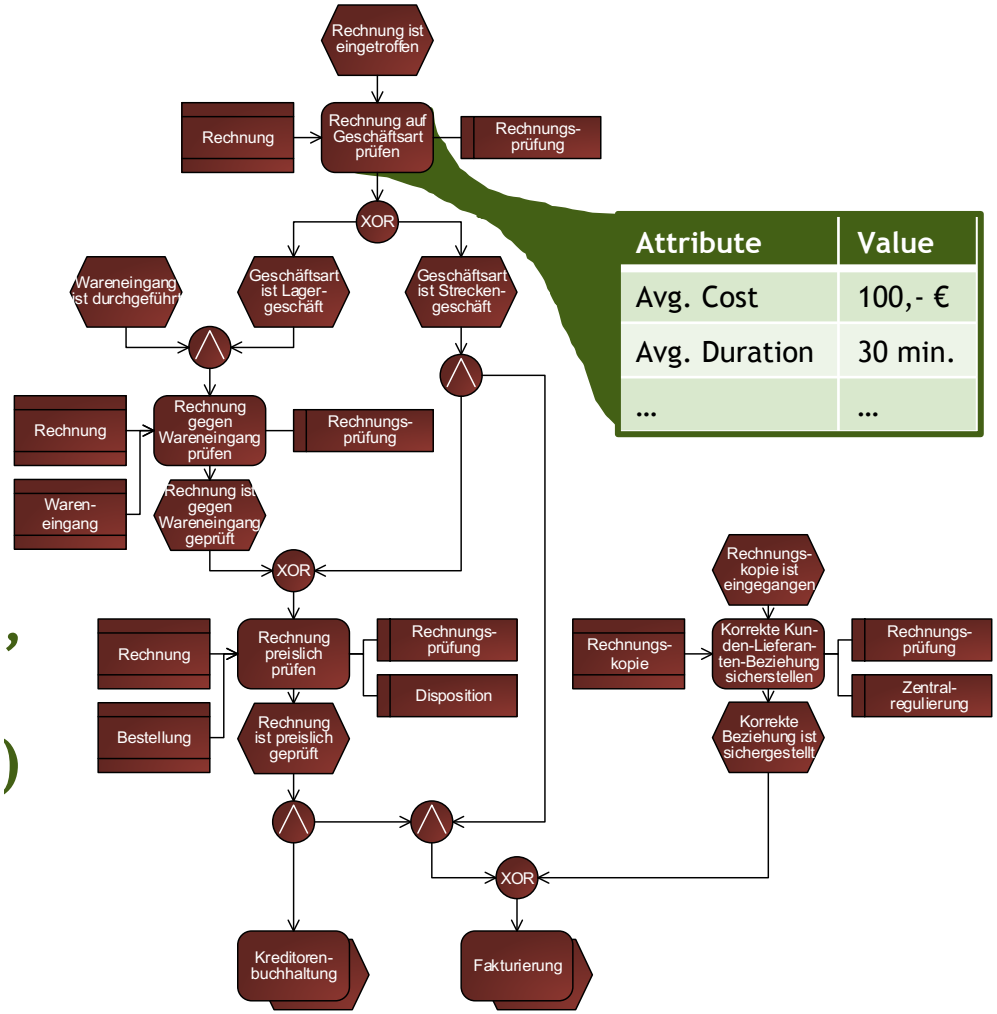
FORMAL REPRESENTATION OF (PROCESS) MODELS

- Vertices
- Edges
- **Properties of Vertices**
  - Vertex Type
  - **Vertex Name**



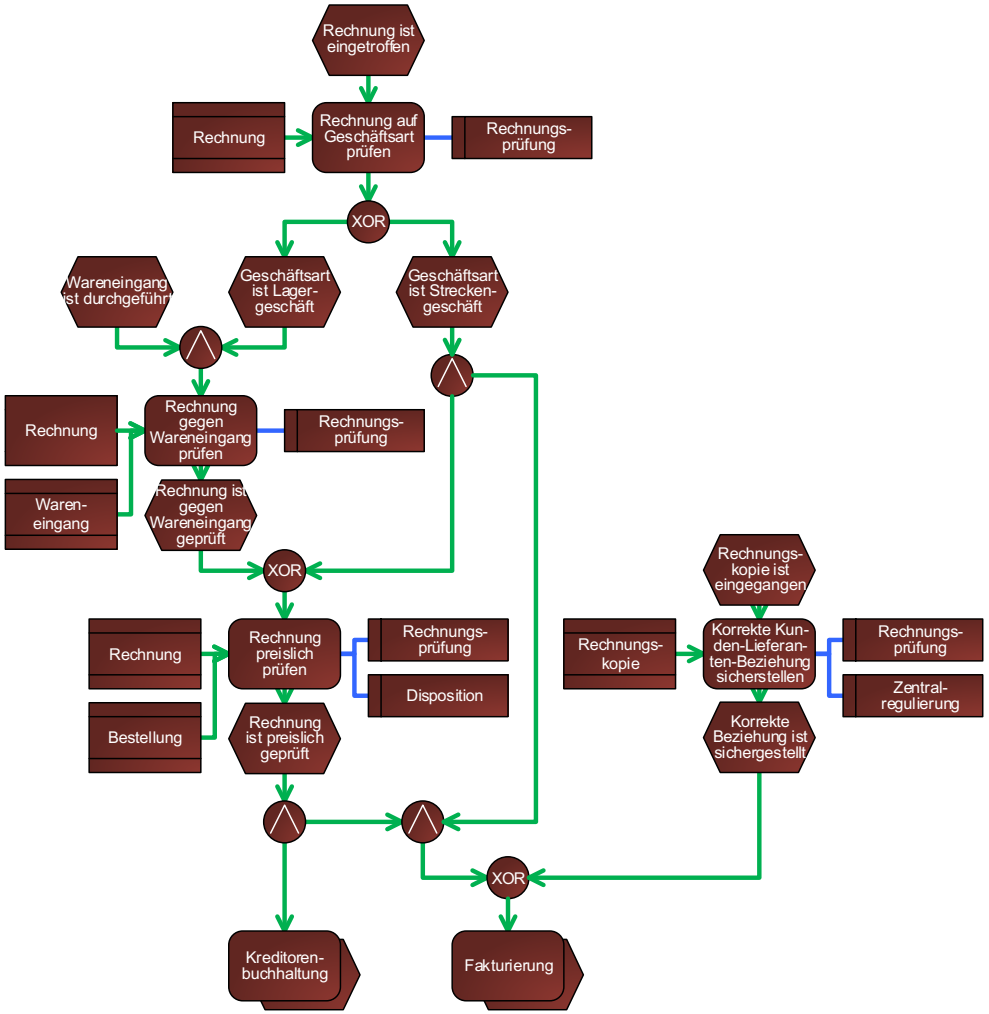
FORMAL REPRESENTATION OF (PROCESS) MODELS

- Vertices
- Edges
- **Properties of Vertices**
  - Vertex Type
  - Vertex Name
  - **Attributes of different kinds (e.g., cost, duration, description, degree of automation, address, etc.)**



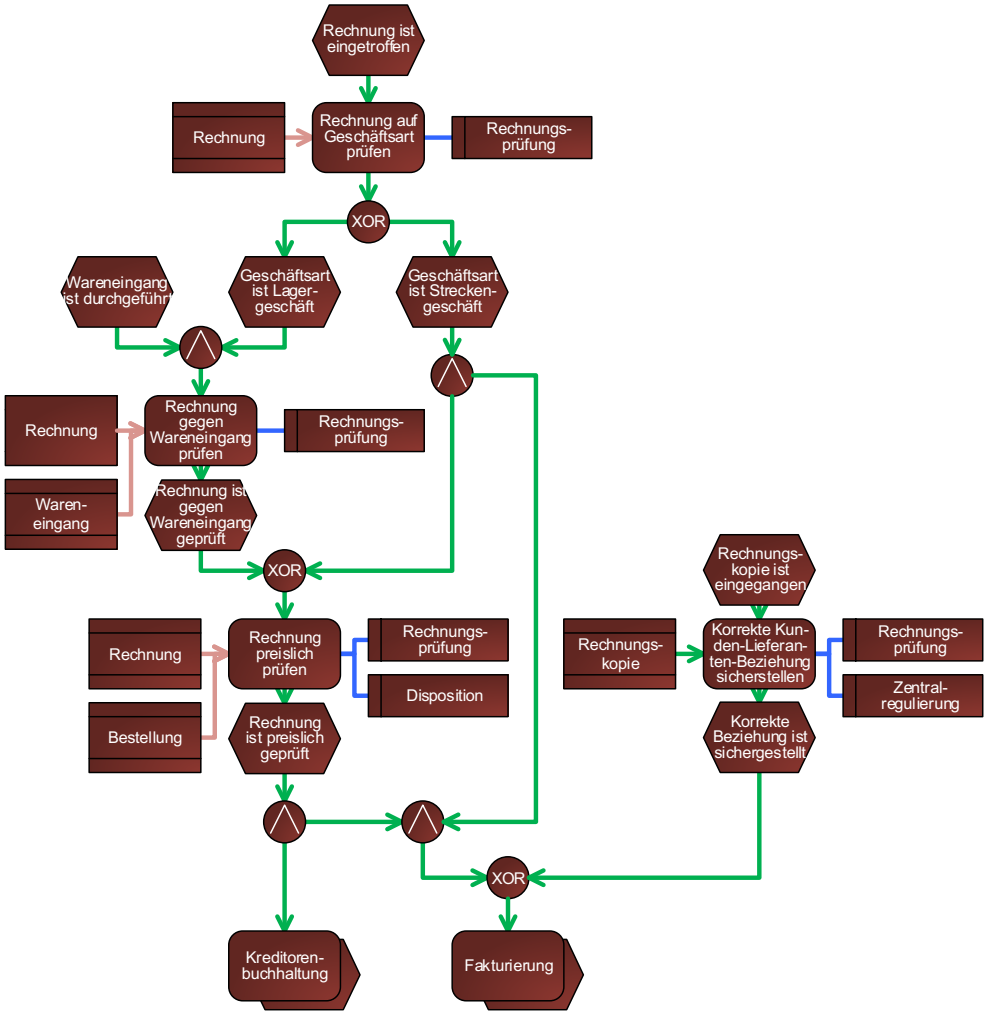
FORMAL REPRESENTATION OF (PROCESS) MODELS

- Vertices
- Edges
- Properties of Vertices
  - Vertex Type
  - Vertex Name
  - Attributes of different kinds (e.g., cost, duration, description, degree of automation, address, etc.)
- **Properties of Edges**
  - **Edge Direction**



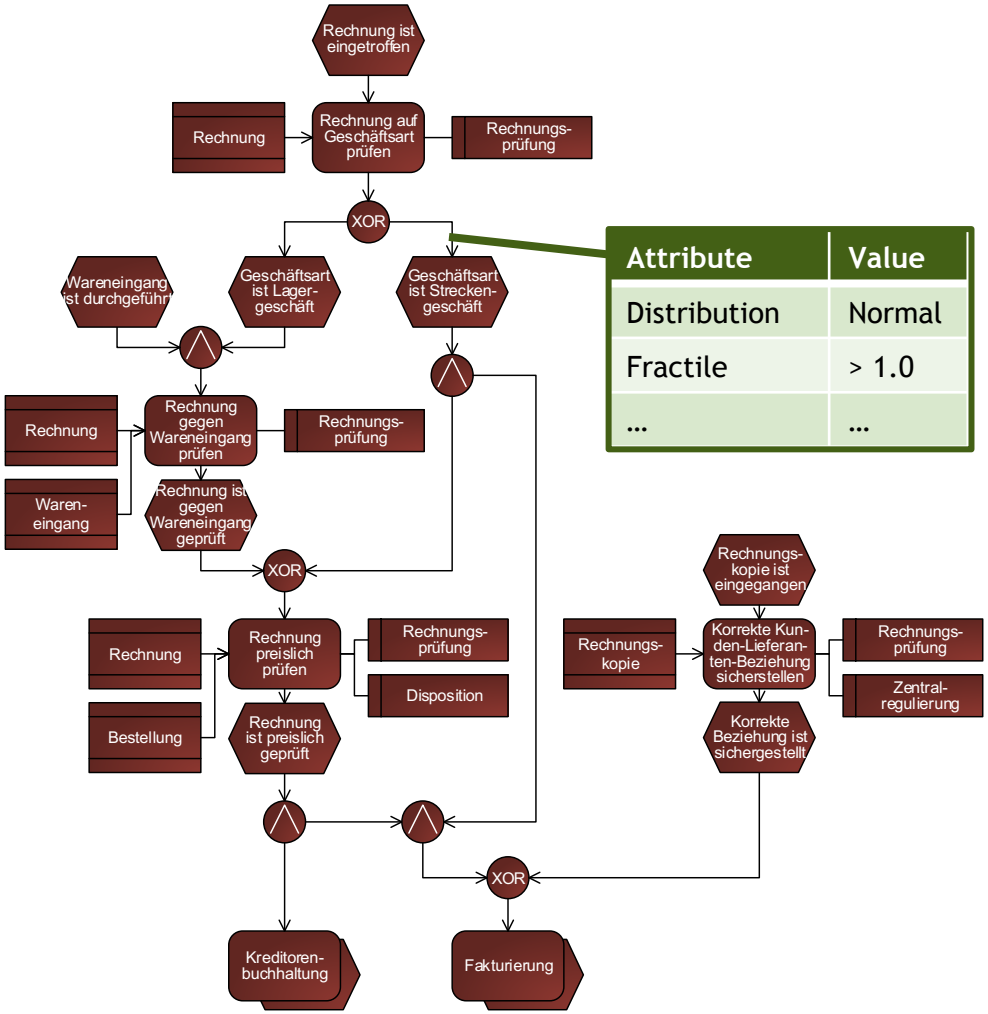
FORMAL REPRESENTATION OF (PROCESS) MODELS

- Vertices
- Edges
- Properties of Vertices
  - Vertex Type
  - Vertex Name
  - Attributes of different kinds (e.g., cost, duration, description, degree of automation, address, etc.)
- **Properties of Edges**
  - Edge Direction
  - **Edge Type**



FORMAL REPRESENTATION OF (PROCESS) MODELS

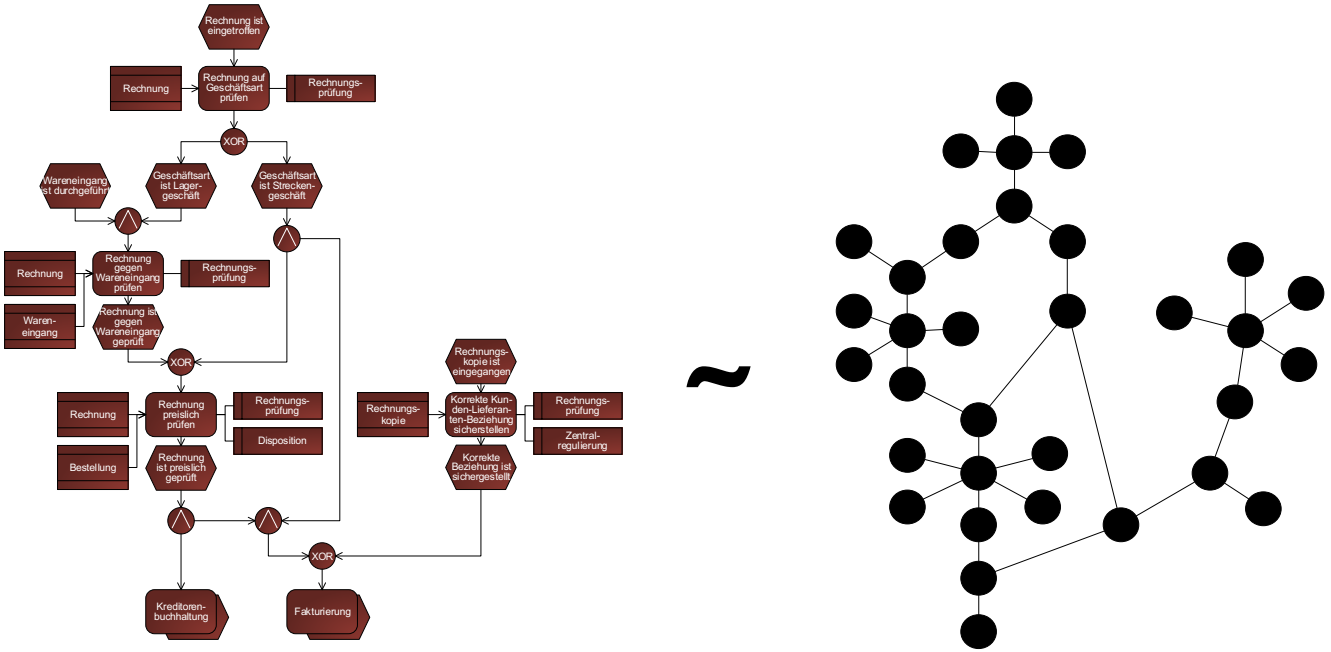
- Vertices
- Edges
- Properties of Vertices
  - Vertex Type
  - Vertex Name
  - Attributes of different kinds (e.g., cost, duration, description, degree of automation, address, etc.)
- **Properties of Edges**
  - Edge Direction
  - Edge Type
  - **Attributes of different kinds**



FORMAL REPRESENTATION OF (PROCESS) MODELS

- A (process) model is a
  - both directed and undirected
  - vertex-attributed and edge-attributed

- GRAPH

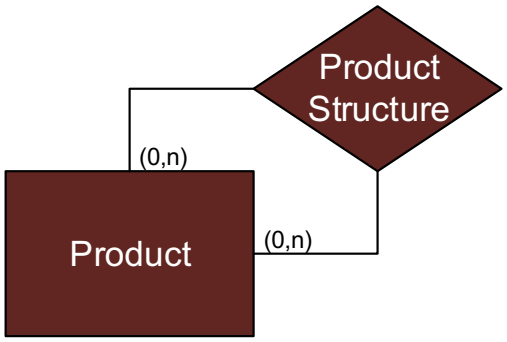


FORMAL REPRESENTATION OF (PROCESS) MODELS

- A (process) model is a
  - both directed and undirected
  - vertex-attributed and edge-attributed

- GRAPH

- Remember: in BPM, we do not need process models only, but also other kinds of models!
- Further requirement: multiple edges between the same nodes (e.g., data models)



FORMAL REPRESENTATION OF (PROCESS) MODELS

- A (process) model is a tuple

$$M=(V,E,C,L,T_V,T_E,\alpha,\beta,\chi)$$

- V is the set of vertices
- E is the set of edges, where $E=E_D \cup E_U$
- $E_D \subseteq V \times V \times \mathbb{N}$ is the set of directed multi-edges
- $E_U \subseteq \{\{v_x,v_y,n\} \mid v_x,v_y \in V, n \in \mathbb{N}\}$ is the set of undirected multi-edges (attention: self-loops)

- We write $Z=V \cup E$ to denote the set of all model elements, including vertices and edges

FORMAL REPRESENTATION OF (PROCESS) MODELS

- A (process) model is a tuple

$$M=(V,E,C,L,T_V,T_E,\alpha,\beta,\chi)$$

- T_V is the set of vertex types (e.g., “activity”, “event”, etc.)
- T_E is the set of edge types (e.g., “sequence flow”, etc.)

- α: $V \rightarrow T_V$ is a function assigning each vertex a type T_V
- β: $E \rightarrow T_E$ is a function assigning each edge a type T_E

- C is a set of captions (e.g., “check bill”, “100,- €”, etc.)
- χ: $Z \rightarrow C$ is a function assigning captions to vertices/edges

FORMAL REPRESENTATION OF (PROCESS) MODELS

- A (process) model is a tuple

$$M=(V,E,C,L,T_V,T_E,\alpha,\beta,\chi)$$

- L is the modeling language the model M belongs to
- This means that L defines, which types of vertices can be connected by which types of edges
- Hence, $T_V,T_E \in L$

FORMAL REPRESENTATION OF (PROCESS) MODELS

- A modeling language is a tuple

$$L=(T_V,T_E)$$

- T_V is the set of vertex types (e.g., “activity”, “event”, “duration”, “cost” etc.)
- T_E is the set of edge types (e.g., “sequence flow”, etc.), where $T_E=T_{ED}\cup T_{EU}$
- We write $T=T_V\cup T_E$ to denote the set of all element types

- $T_{ED}\subseteq T_V\times T_V\times\mathbb{N}$ is the set of directed edge types
- $T_{EU}\subseteq\{\{t_{vx},t_{vy},n\} \mid t_{vx},t_{vy}\in T_V,\ n\in\mathbb{N}\}$ (attention: self-loops) is the set of undirected edge types

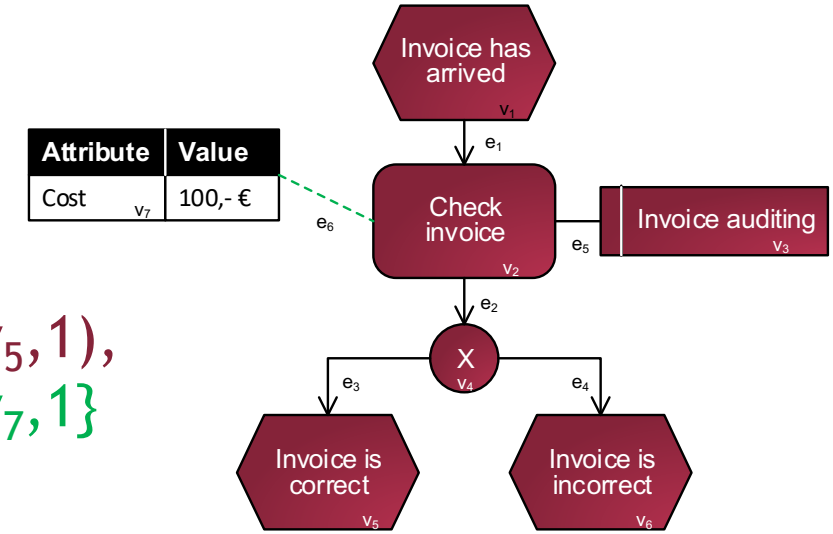
EXAMPLE: EPC: LANGUAGE

- T_V={function,event,and,xor,or,orgunit,cost,name}
- T_{ED}={f_e,f_a,f_x,f_o,e_f,e_a,e_x,e_o,a_f,a_e,a_a,a_x, a_o, x_f,x_e,x_a,x_x,x_o,o_f,o_e,o_a,o_x,o_o}
- T_{EU}={f_c,f_org,f_n,e_n,org_n}

- Examples of T_{ED}: f_e=(function,event,1), x_x=(xor,xor,1)
- Example of T_{EU}: f_org={function,orgunit,1}

EXAMPLE: EPC MODEL

WITH CAPTIONS STRAIGHTLY ANNOTATED TO VISIBLE VERTICES



- $V=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7\}$
- $E_D=\{e_1,e_2,e_3,e_4\}$
- $E_U=\{e_5,e_6\}$
- $e_1=(v_1,v_2,1)$, $e_2=(v_2,v_4,1)$, $e_3=(v_4,v_5,1)$, $e_4=(v_4,v_6,1)$, $e_5=\{v_2,v_3,1\}$, $e_6=\{v_6,v_7,1\}$

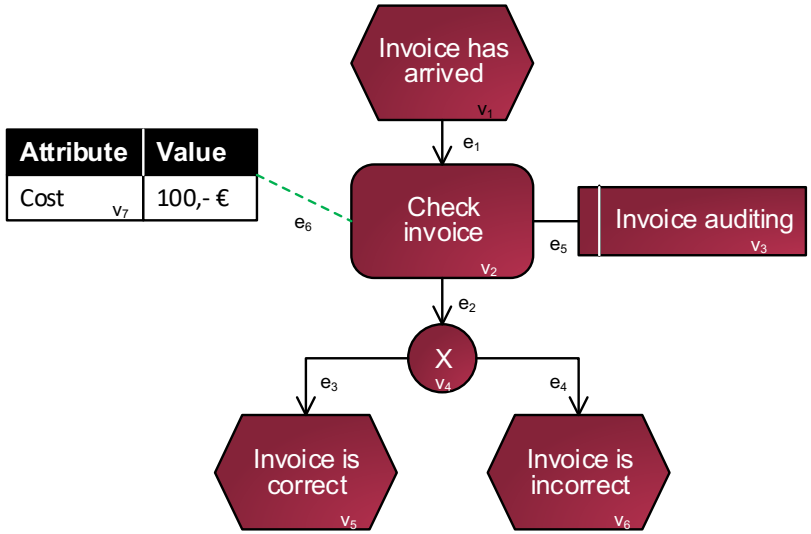
- $T_V=T_V(EPC)$, $T_E=T_E(EPC)$

- $\alpha(v_1)=\alpha(v_5)=\alpha(v_6)=$event, $\alpha(v_2)=$function, $\alpha(v_4)=$XOR, $\alpha(v_3)=$orgunit, $\alpha(v_7)=$cost
- $\beta(e_1)=e\_f$, $\beta(e_2)=f\_x$, $\beta(e_3)=\beta(e_4)=x\_e$, $\beta(e_5)=f\_org$, $\beta(e_6)=f\_c$

EXAMPLE: EPC MODEL

WITH CAPTIONS STRAIGHTLY ANNOTATED TO VISIBLE VERTICES

- $\chi(v_1)$="Invoice has arrived"
- $\chi(v_2)$="Check invoice"
- $\chi(v_3)$="Invoice auditing"
- $\chi(v_5)$="Invoice is correct"
- $\chi(v_6)$="Invoice is incorrect"
- $\chi(v_7)$="100,- €"



- Note that $\chi(v_4)$ and $\chi(e_i)$ $\forall$ i={1..6} are not defined, i.e., those vertices and edges do not carry any value

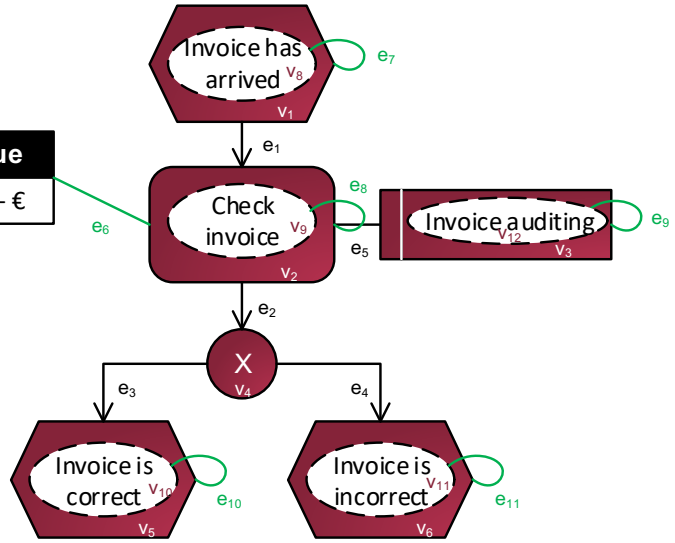
WE JUST ASSIGNED LABELS DIRECTLY TO VERTICES ETC.

- Most modeling tools do not do this
- The reason is that we treat every attribute of a model element equally, i.e. a <<name>> is “worth” as much as <<cost>> or the like.

EXAMPLE: EPC: MODEL

- $V=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{10},v_{11},v_{12}\}$
- $E_D=\{e_1,e_2,e_3,e_4\}$
- $E_U=\{e_5,e_6,e_7,e_8,e_9,e_{10},e_{11}\}$
- $e_1=(v_1,v_2,1)$, $e_2=(v_2,v_4,1)$, $e_3=(v_4,v_5,1)$, $e_4=(v_4,v_6,1)$, $e_5=\{v_2,v_3,1\}$, $e_6=\{v_2,v_7,1\}$, $e_7=\{v_1,v_8,1\}$, $e_8=\{v_2,v_9,1\}$, $e_9=\{v_3,v_{12},1\}$, $e_{10}=\{v_5,v_{10},1\}$, $e_{11}=\{v_6,v_{11},1\}$

Attribute	Value
Cost v_7	100,- €

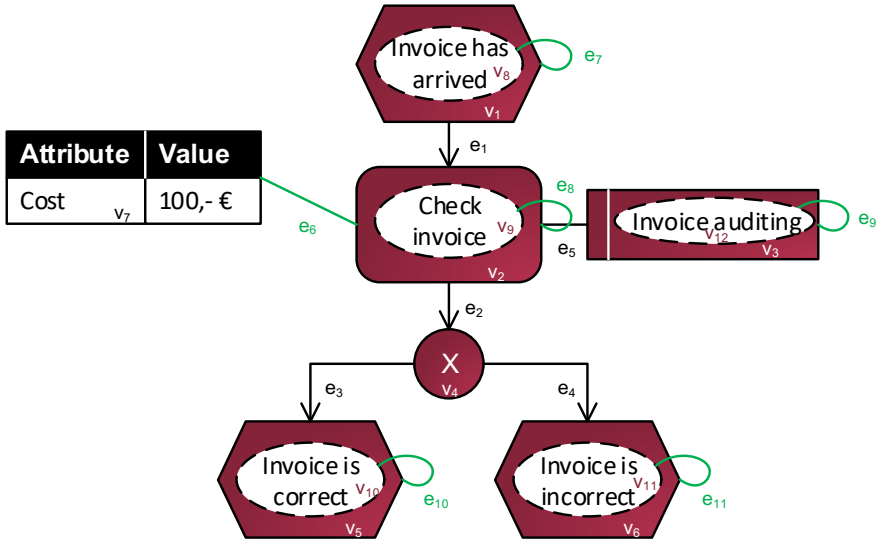


- $T_V=T_V(EPC)$, $T_E=T_E(EPC)$

- $\alpha(v_1)=\alpha(v_5)=\alpha(v_6)=$event, $\alpha(v_2)=$function, $\alpha(v_4)=$XOR, $\alpha(v_3)=$orgunit, $\alpha(v_7)=$cost, $\alpha(v_8)=\alpha(v_9)=\alpha(v_{10})=\alpha(v_{11})=\alpha(v_{12})=$name
- $\beta(e_1)=$e_f, $\beta(e_2)=$f_x, $\beta(e_3)=\beta(e_4)=$x_e, $\beta(e_5)=$f_org, $\beta(e_6)=$f_c, $\beta(e_7)=\beta(e_{10})=\beta(e_{11})=$e_n, $\beta(e_8)=$f_n, $\beta(e_9)=$org_n

EXAMPLE: EPC: MODEL

- $\chi(v_8)$="Invoice has arrived"
- $\chi(v_9)$="Check invoice"
- $\chi(v_{12})$="Invoice auditing"
- $\chi(v_{10})$="Invoice is correct"
- $\chi(v_{11})$="Invoice is incorrect"
- $\chi(v_7)$="100,- €"



- Note that $\chi(v_i)\ \forall\ i=\{1..6\}$ and $\chi(e_j)\ \forall\ j=\{1..11\}$ are not defined, i.e., those vertices and edges do not carry any value

AGENDA

- Formalization of (Process) Models
- Storing (Process) Models

EXAMPLE: A META MODELING TOOL DATA MODEL

- The data model must be able to store
  - Languages (→ model types)
  - Language objects (→ vertex types)
  - Language element relationships (→ edge types)
  - Models
  - Model vertices
  - Model edges

STORING MODELS IN MODELING TOOLS

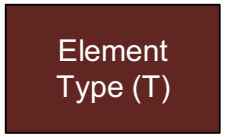
PRELIMINARY CONSIDERATIONS

- To access a conceptual model by some automatism performing BPM tasks, we have to store it
- Database
- A data model of how to store conceptual models

- We must be able to store information about
  - the models
  - their languages
  - i.e., M and L (see 1st chapter)

STORING MODELS IN MODELING TOOLS

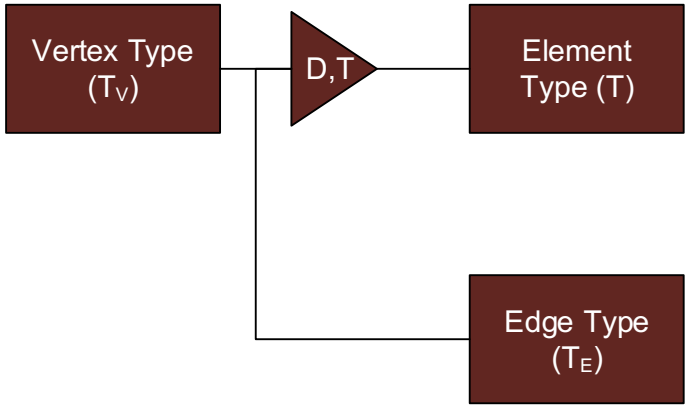
DATABASE



STORING MODELS IN MODELING TOOLS

DATABASE

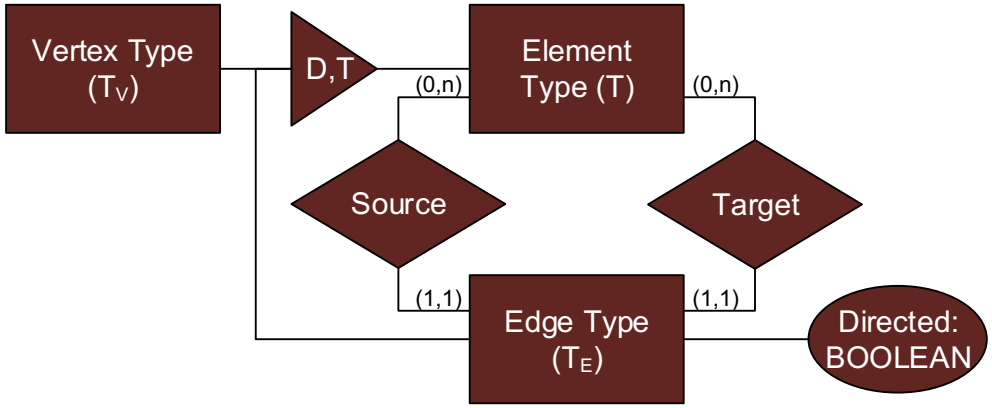
- $T=T_V \cup T_E$



STORING MODELS IN MODELING TOOLS

DATABASE

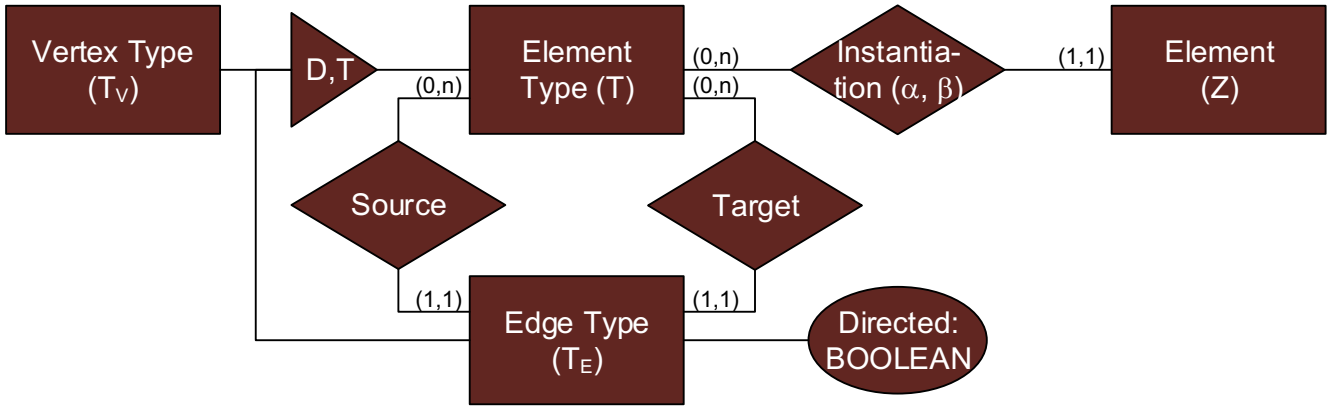
- $T = T_V \cup T_E$
- $T_{ED} \subseteq T_V \times T_V \times \mathbb{N}$
- $T_{EU} \subseteq \{\{t_{vx}, t_{vy}, n\} \mid t_{vx}, t_{vy} \in T_V, n \in \mathbb{N}\}$



STORING MODELS IN MODELING TOOLS

DATABASE

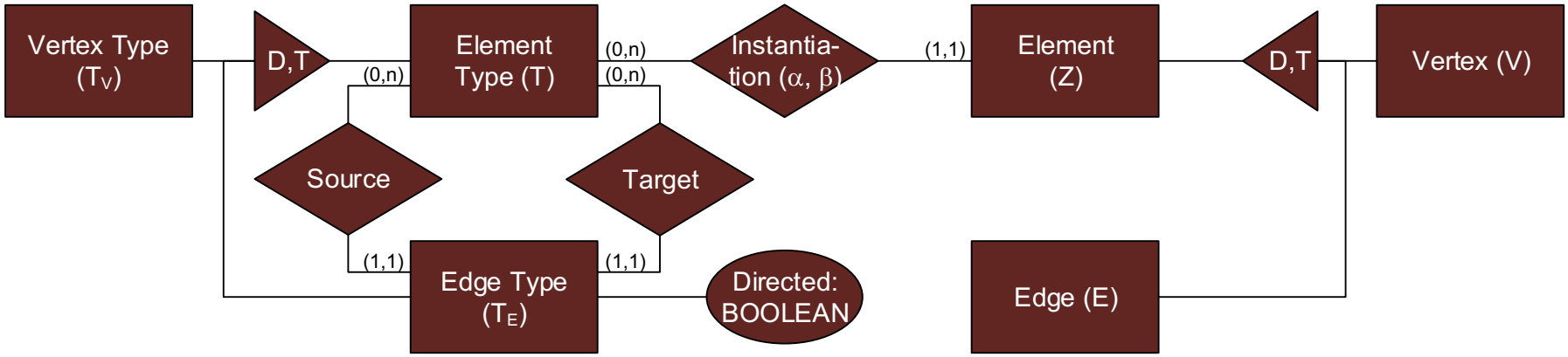
- α: $V \rightarrow T_V$
- β: $E \rightarrow T_E$



STORING MODELS IN MODELING TOOLS

DATABASE

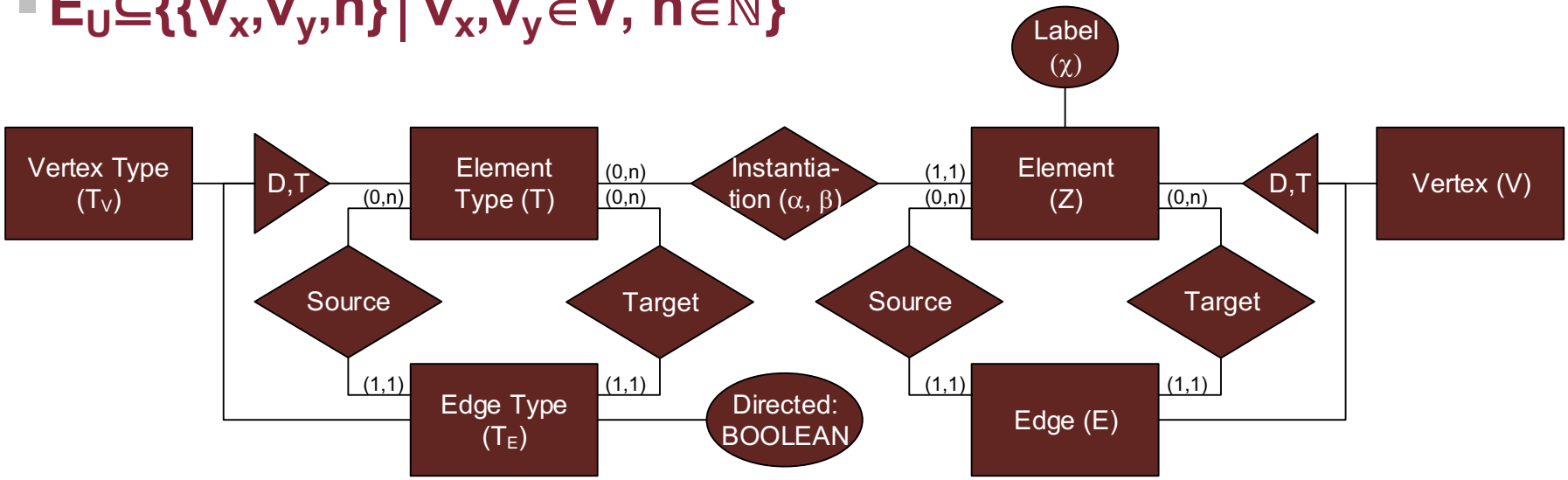
- $Z = V \cup E$



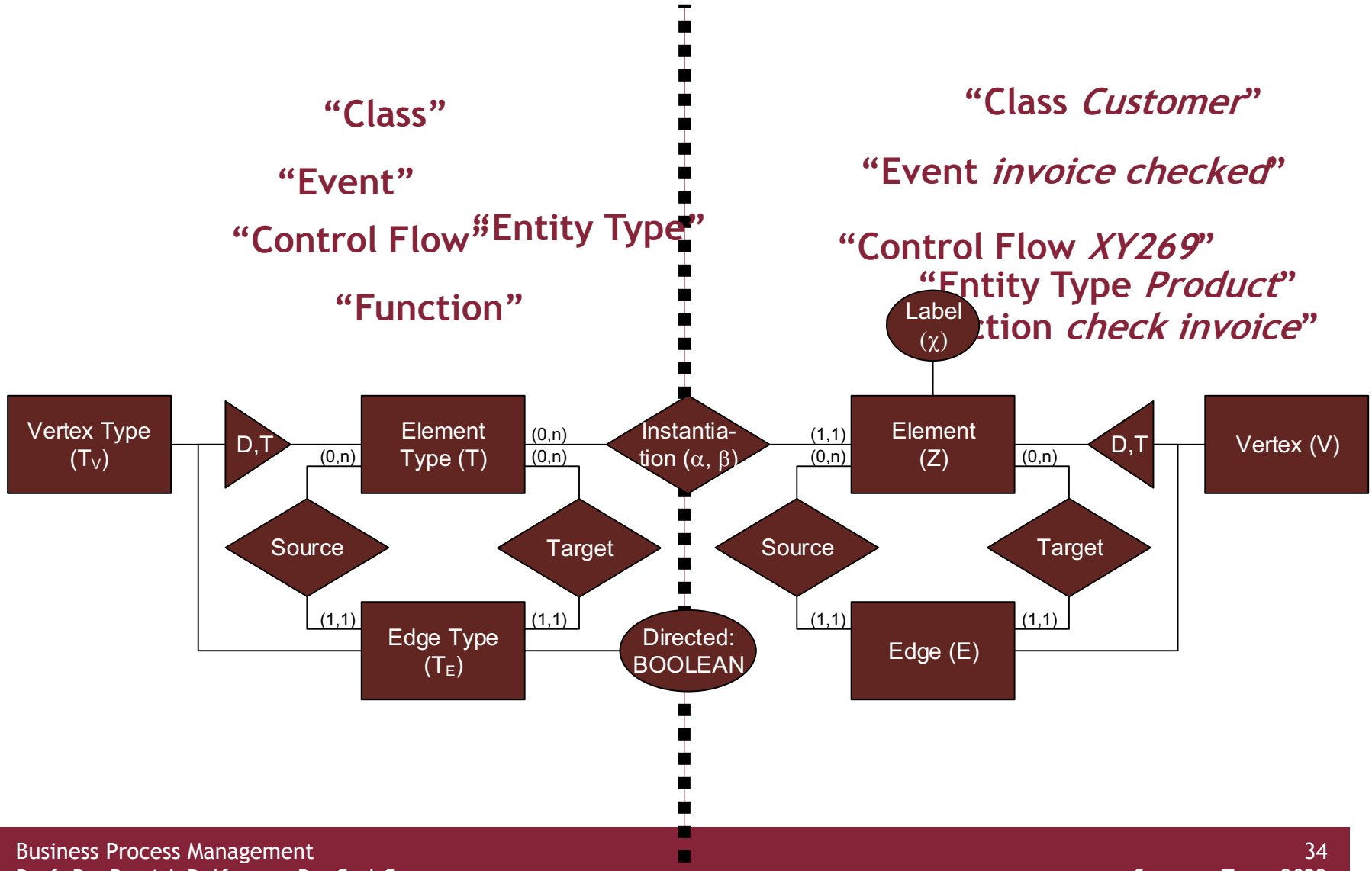
STORING MODELS IN MODELING TOOLS

DATABASE

- $E=E_D \cup E_U$
- $E_D \subseteq V \times V \times \mathbb{N}$
- $E_U \subseteq \{\{v_x, v_y, n\} \mid v_x, v_y \in V,\ n \in \mathbb{N}\}$



- $\chi: Z \rightarrow C$


"Class"
"Event"
"Control Flow"
"Function"
"Entity Type"
"Class *Customer*"
"Event *invoice checked*"
"Control Flow *XY269*"
"Entity Type *Product*"
"tion *check invoice*"
Label (χ)
Vertex Type (T_V)
D,T
(0,n)
Element Type (T)
(0,n)
(0,n)
Instantia-tion (α, β)
(1,1)
(0,n)
Element (Z)
(0,n)
D,T
Vertex (V)
Source
Target
Source
Target
(1,1)
(1,1)
(1,1)
(1,1)
Edge Type (T_E)
Directed: BOOLEAN
Edge (E)

UTILITY

- Models are stored in a technical way
- Model data can be accessed automatically
- Designers of BPM algorithms can rely on an unambiguous specification

UNIVERSITÄT KOBLENZ · LANDAU

IWVI

BUSINESS PROCESS MANAGEMENT

FORMALIZATION OF (PROCESS) MODELS AND (PROCESS) MODELING TOOLS

INSTITUTE FOR IS RESEARCH

www.uni-koblenz.de