Written Examination of the Lecture
Business Process Management (BPM)
Prof. Dr. Patrick Delfmann
Summer Term 2019, 2019-10-10, 10:00h, G 309

Matriculation number: _____

Program of study: _____

Forename and surname: _____

Please note your name on **each** page.

Duration: 90 minutes
Maximum points: 90

Task		Points
Task 1: Formalization of (Process) Models	(15 Points)	
Task 2: Model Query with CTL	(12 Points)	
Task 3: Model Query with GMQL	(26 Points)	
Task 4: Model Query with DMQL	(10 Points)	
Task 5: Business Rules	(27 Points)	
Sub-total	(90 Points)	
Bonus points from the Exercise	(+10 Points)	
Total		
Final Grade		

Name: _____

Task 1: Formalization of (Process) Models (15 Points)

Below you find a formalization of an EPC model based on the notion of conceptual model as introduced in the lecture:

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

Turn the following formalization into a corresponding graphical representation of an EPC (*hint: here, we assume that we can assign captions **directly** to vertices and edges without having to specify extra caption vertices*):

$V=\{v_1,...,v_{10}\}$, $E=\{e_1,..., e_9\}$

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

C={"Goods receipt", "Check goods", "Goods released", "Goods blocked", "Goods rejected", "Order", "Delivery note", "Inspection result", "Receiving department"}

$L=EPC=(T_V,T_E)$

$T_V=\{$***e***vent, ***f***unction, ***x***or, ***d***ata, ***o***rganization$\}$, $T_E=\{e \rightarrow f, f \rightarrow x, x \rightarrow e, d \rightarrow f, f \rightarrow d, f—o\}$

$\alpha(v_1)=\alpha(v_4)=\alpha(v_5)=\alpha(v_6)=$event, $\alpha(v_2)=$function, $\alpha(v_3)=$xor, $\alpha(v_7)=\alpha(v_8)=\alpha(v_9)=$data, $\alpha(v_{10})=$organization

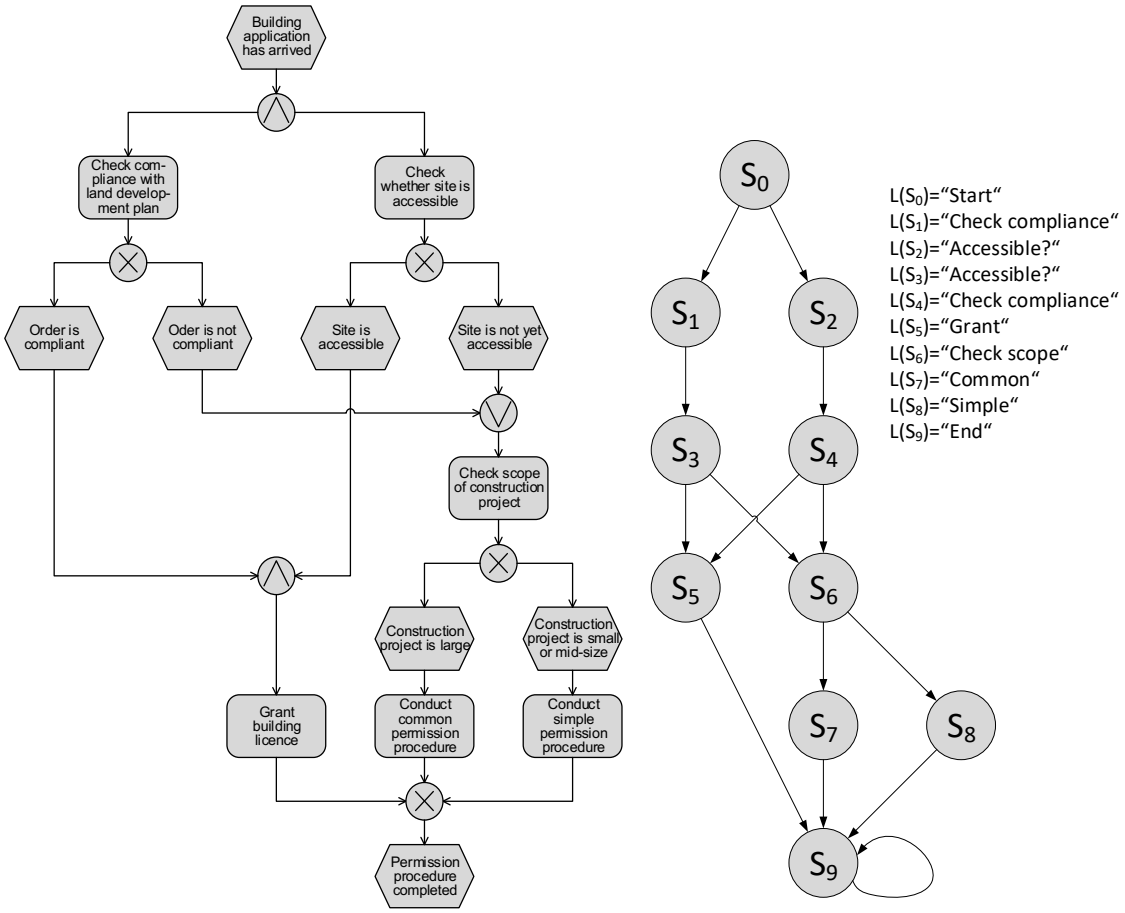
$\beta(e_1)=e \rightarrow f$, $\beta(e_2)=f \rightarrow x$, $\beta(e_3)=\beta(e_4)=\beta(e_5)=x \rightarrow e$, $\beta(e_6)=\beta(e_7)=d \rightarrow f$, $\beta(e_8)=f \rightarrow d$, $\beta(e_9)=f—o$

$\chi(v_1)=$"Goods receipt", $\chi(v_2)=$"Check goods", $\chi(v_4)=$"Goods released", $\chi(v_5)=$"Goods blocked", $\chi(v_6)=$"Goods rejected", $\chi(v_7)=$"Order", $\chi(v_8)=$"Delivery note" , $\chi(v_9)=$" Inspection result", $\chi(v_{10})=$"Receiving department", $\chi(v_3)=\chi(e_i)=\varnothing \; \forall i \in \{1..9\}$

Name: _____

Task 2: Model Query with CTL (12 Points)

The figure below shows an EPC describing the building application process of public administrations. To examine it with CTL, it has already been transformed into a transition system also shown in the figure.



Develop CTL queries that check the transition system for the aspects specified below. Your queries should all start in the beginning state. You can use the following statements to assemble your queries:

***A** ("for all paths"), **E** ("there exists a path"), **X** ("in the next state"), **F** ("in a future state"), **G** ("globally in the future"), **U** ("until")*

The query must obey the following CTL syntax, where p is an atomic proposition:

$$\varphi ::= p \mid \neg\varphi \mid \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \varphi\rightarrow\varphi \mid AX\varphi \mid EX\varphi \mid AF\varphi \mid EF\varphi \mid AG\varphi \mid EG\varphi \mid A[\varphi U\varphi] \mid E[\varphi U\varphi]$$

Build queries that check the transition system for the following aspects:

a) The process must provide both a simple permission procedure and a common permission procedure. (3 points)

b) The building license can only be granted when both the compliance with the development plan and the accessibility of the building site have been checked before. (7 points)

c) A grant can only be given at the end of the permission process. (2 points)

Task 3: Model Query with GMQL (26 Points)

Please consider the given process model in EPC notation. All vertices in the model are numbered using the prefix "v" (v1 to v35), and all edges are numbered using the prefix "e" (e1 to e36). Please use these identifiers when referring to vertices or edges from the model. Below, you can find a table with all GMQL functions and their parameters. Note that the curly brackets provided in the operations list mean that there are multiple operations. E.g., "{Directed}Paths" means that there is an operation called "Paths" and there is another operation called "DirectedPaths".

Basic Sets

V: the set of all vertices available; E: the set of all edges available
Z: the set of all elements available; $Z=V \cup E$
T_V: the set of all vertex types available;
$T_V=\{Function, Event, XOR, OR, AND, OrgUnit, Data\}$
T_E: the set of all edge types available; $T_E=\{ControlFlow, Responsible, Input, Output\}$

- *ControlFlows* connect *Functions*, *Events*, *XOR*, *OR*, and *AND* in any manner, except *Function*→*Function* and *Event*→*Event*.
- *Responsible* connects *Functions* with *OrgUnits*.
- *Input*: Data→Function; *Output*: Function→Data
- *directed(ControlFlow)=directed(Input)=directed(Output)=TRUE* and *directed(Responsible)=FALSE*

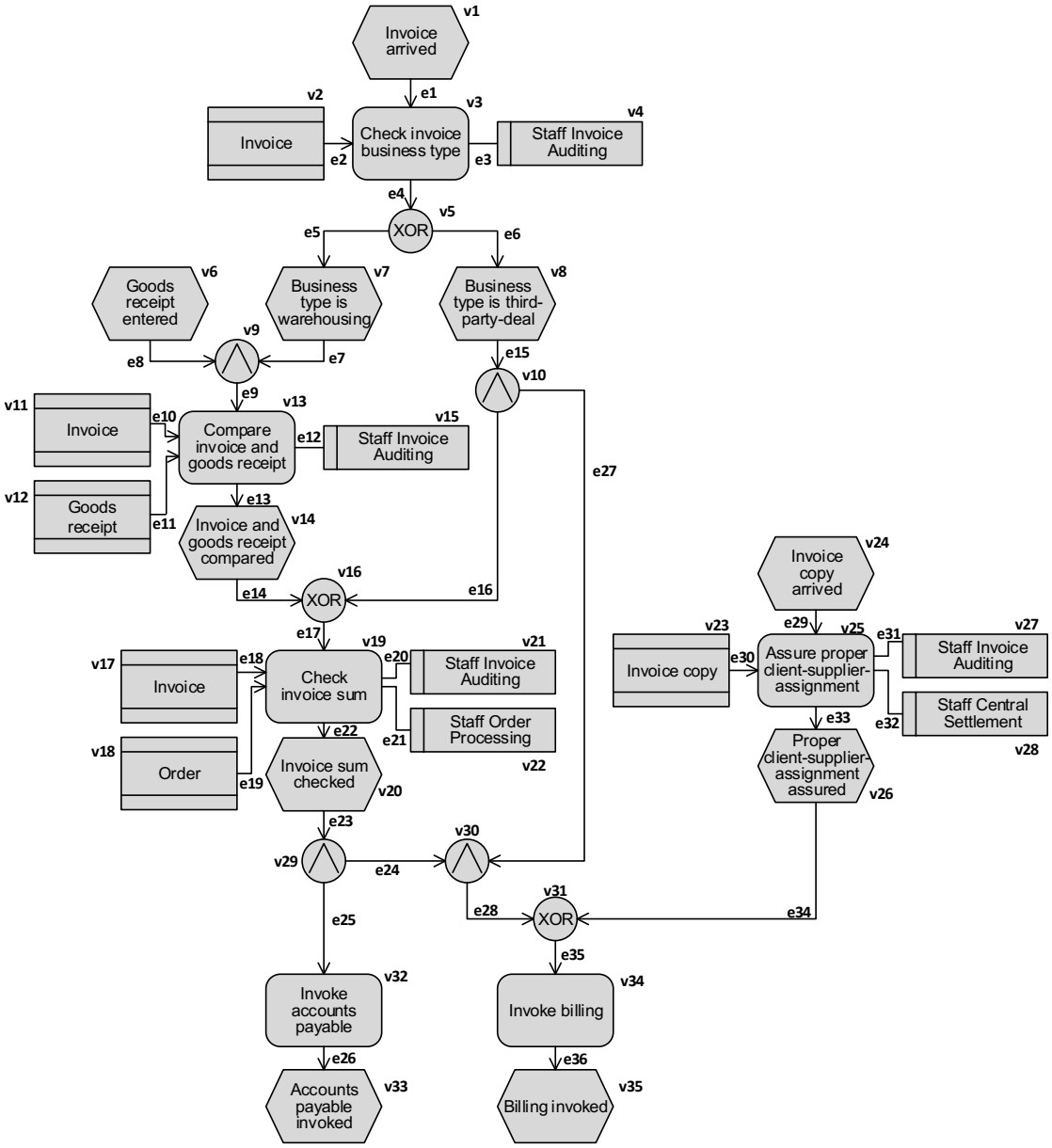
T: the set of all element types available; $T=T_V \cup T_E$; t is a particular element type
X_i: an arbitrary set of elements; Y_V: an arbitrary set of vertices; Y_E: an arbitrary set of edges
n_x: a natural number; *value*: a String

Set Operations	Abbreviation
ElementsOfType(X,t)	*EOT*
ElementsWithAttributeOfValue(X,t,value)	*EWAOV*
ElementsWithAttributeOfDataType(X,value)	*EWAODT*
ElementsWith{Pred|Succ}Relations(X,Y_E)	*EW{P|S}R*
ElementsWith{Pred|Succ}RelationsOfType(X,Y_E,t_E)	*EW{P|S}ROT*
ElementsWithNumberOf{Pred|Succ}Relations(X,n_x)	*EWN{P|S}R*
ElementsWithNumberOf{Pred|Succ}RelationsOfType(X,t_E,n_x)	*EWN{P|S}ROT*
ElementsDirectlyRelated(X_1,X_2)	*EDR*
AdjacentSuccessors(X_1,X_n)	*AS*
{Directed}Paths(X_1,X_n)	*{D}P*
{Directed}PathsContainingElements(X_1,X_n,X_c)	*{D}PCE*
{Directed}PathsNotContainingElements(X_1,X_n,X_c)	*{D}PNCE*
{Directed}Loops(X)	*{D}L*
{Directed}LoopsContainingElements(X,X_c)	*{D}LCE*
{Directed}LoopsNotContainingElements(X,X_c)	*{D}LNCE*

Set Operators

UNION, JOIN, INTERSECTION, INNER_INTERSECTION, COMPLEMENT, INNER_COMPLEMENT, SELF_UNION, SELF_INTERSECTION

Name: _____



Please provide the result sets for the following GMQL queries, when they are executed against the given EPC model (see next page). For example, the query EOT(V, XOR) would return the set $\{v_5, v_{16}, v_{31}\}$. *Hint: please consider that some GMQL queries return* ***sets****, whereas other GMQL queries return* ***sets of sets****. If the result contains sequences of vertices or edges, you can use abbreviations like* $v_1,\ldots,v_n$.

a) EOT(V, AND) (1.5 points)

b) EWAOV(E, Caption, "*") (1.5 points)

c) DP(
SELF_UNION(EWNOPR(EOT(V,Event),0)),
SELF_UNION(EWNOSR(EOT(V,Event),0))
) (8 points)

d) JOIN(
AS(
EWAOV(EOT(V,Data),Caption,"Invoice*"),
EOT(V,Function)
),
EDR(
EOT(V,Function),
EOT(V,OrgUnit)
)
) (8 points)

Now think the other way round. Please provide a GMQL query, which leads to the given result set. Obviously, there are many possible solutions – you only need to specify one single correct one.

e) { {v_6, e_8, v_9, e_9, v_{13}, e_{13}, v_{14}, e_{14}, v_{16}, e_{17}, v_{19}, e_{22}, v_{20}, e_{23}, v_{29}, e_{25}, v_{32}, e_{26}, v_{33}} } (4 points)

f) {v_2, v_4, v_{11}, v_{12}, v_{15}, v_{17}, v_{18}, v_{21}, v_{22}, v_{23}, v_{27}, v_{28}} (3 points)

Name: _____

Task 4: Model Query with DMQL (10 Points)

a) Build a query with DMQL that returns the same results as the GMQL query from task (3c) or, in case it is not possible, explain why. *Hint: Include all relevant specifications in your description. This also includes global rules, if necessary.* (6 points)

b) Build a query with DMQL that returns the same results as the GMQL query from task (3d) or, in case it is not possible, explain why. *Hint: Include all relevant specifications in your description. This also includes global rules, if necessary.* (4 points)

Name: _____

Task 5: Business Rules (27 Points)

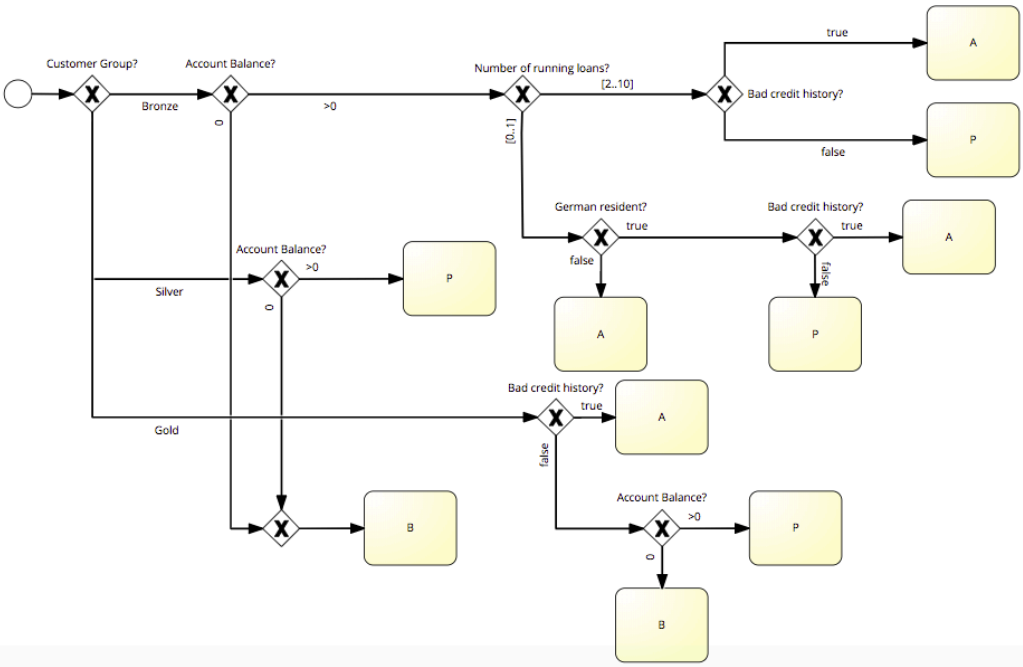
a) Consider the following DECLARE rule base. The shown rule base contains contradictions. Define the minimal contradictory subsets for the shown rule base as introduced in the lecture. *Hint: You do **not** need to draw the minimal contradictory subsets graphically.* (10 points)

ChainResponse(a,b)
NotPrecedence(e,b)
NotPrecedence(b,e)
NotResponse(a,h)
Response(d,g)
Response(f,h)
Precedence(b,e)
ChainResponse(b,d)
Response(d,f)
Response(g,h)
NotResponse(b,d)

b) Which constraint(s) from Task (a) has/have the highest degree of culpability? (1 point)

c) How many distinct constraints from Task (a) are part of the contradictions? (1 point)

d) Consider the following BPMN diagram. It is an exemplary diagram for determining the risk status (a code from A-H) for a new customer loan application in a fraud management department of a financial vendor. There seems to be a lot of decision logic integrated into the model on how the risk status is determined (see the respective XOR gateways). Please abstract the shown decision logic into a new DMN Decision Table (Decision Model and Notation format). The resulting DMN Decision Table **must not** contain any errors as discussed in the lecture, i.e. there must be **no** identical rules, **no** overlapping rules and **no** subsumed rules. Your DMN Decision Table **must** contain column types and a hit indicator. Then, re-model the shown BPMN diagram by introducing a business rule task, which is annotated with the DMN decision table you created (this model should contain **no** gateways). *Hint 1: If possible, you can simplify the original (integrated) decision logic when you transform it into a DMN decision table. Hint 2: Your resulting DMN Decision Table only needs to capture the process model "as-is", i.e., you can make use of DMN Hit-Policies and rule order.* (15 points)



Name: _____

Name: _____

Good speed!

Written Examination of the Lecture
Business Process Management (BPM)
Prof. Dr. Patrick Delfmann
Summer Term 2019, 2019-07-18, 14:00h, G 309

Matriculation number: _____

Program of study: _____

Forename and surname: _____

Please note your name on **each** page.

Duration: 90 minutes
Maximum points: 90

Task		Points
Task 1: Formalization of (Process) Models	(15 Points)	
Task 2: Model Query with CTL	(12 Points)	
Task 3: Model Query with GMQL	(26 Points)	
Task 4: Model Query with DMQL	(10 Points)	
Task 5: Business Rules	(27 Points)	
Sub-total	(90 Points)	
Bonus points from the Exercise	(+10 Points)	
Total		
Final Grade		

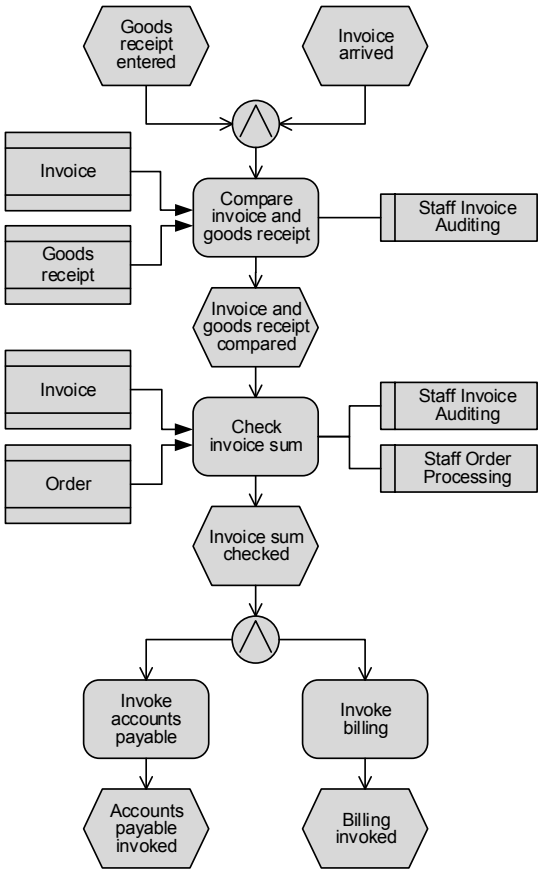
Name: _____

Task 1: Formalization of (Process) Models (15 Points)

The EPC shown on the right depicts an invoice auditing process of classical warehouse retailing. Provide a formalization of the EPC as introduced in the lecture. Use the following tuple as a formalization basis:

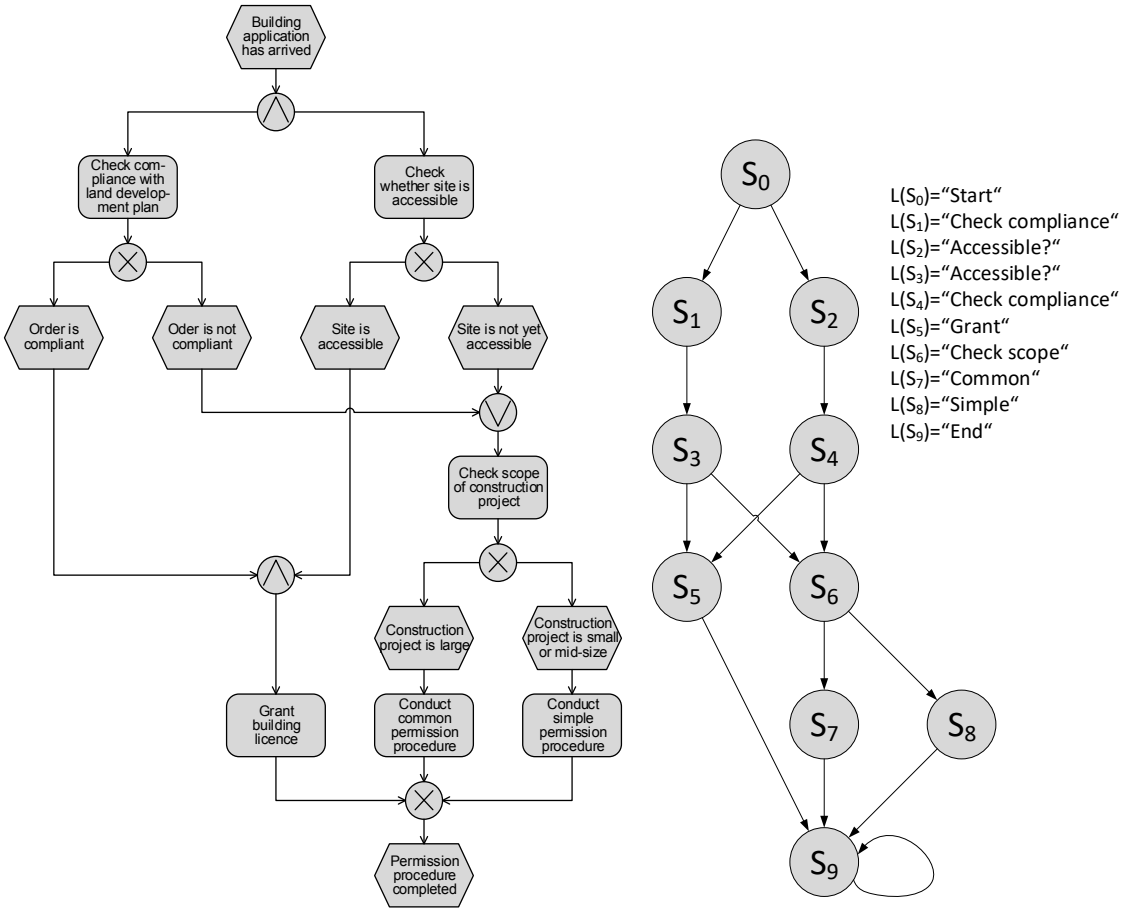
$$M = (V, E, C, L, T_V, T_E, \alpha, \beta, \gamma)$$

Hint: Your formalization only needs to depict this exact EPC, so you do not have to specify a full-fledged eEPC language. Furthermore, you can assume that you can assign captions directly to vertices and edges of any kind.



Task 2: Model Query with CTL (12 Points)

The figure below shows an EPC describing the building application process of public administrations. To examine it with CTL, it has already been transformed into a transition system also shown in the figure.



Develop a CTL query that checks the transition system for the following aspect: *The compliance check with the development plan and the accessibility check have to be performed concurrently (i.e., in parallel)*. Your query should start in the beginning state. You can use the following statements to assemble your query:

***A** ("for all paths"), **E** ("there exists a path"), **X** ("in the next state"), **F** ("in a future state"), **G** ("globally in the future"), **U** ("until")*

The query must obey the following CTL syntax, where p is an atomic proposition:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid AX\varphi \mid EX\varphi \mid AF\varphi \mid EF\varphi \mid AG\varphi \mid EG\varphi \mid A[\varphi U \varphi] \mid E[\varphi U \varphi]$$

Name: _____

Name: _____

Task 3: Model Query with GMQL (26 Points)

Please consider the given process model in EPC notation. All vertices in the model are numbered using the prefix "v" (v1 to v35), and all edges are numbered using the prefix "e" (e1 to e36). Please use these identifiers when referring to vertices or edges from the model. Below, you can find a table with all GMQL functions and their parameters. Note that the curly brackets provided in the operations list mean that there are multiple operations. E.g., "{Directed}Paths" means that there is an operation called "Paths" and there is another operation called "DirectedPaths".

Basic Sets

V: the set of all vertices available; E: the set of all edges available
Z: the set of all elements available; $Z=V \cup E$
T_V: the set of all vertex types available;
$T_V=\{Function, Event, XOR, OR, AND, OrgUnit, Data\}$
T_E: the set of all edge types available; $T_E=\{ControlFlow, Responsible, Input, Output\}$

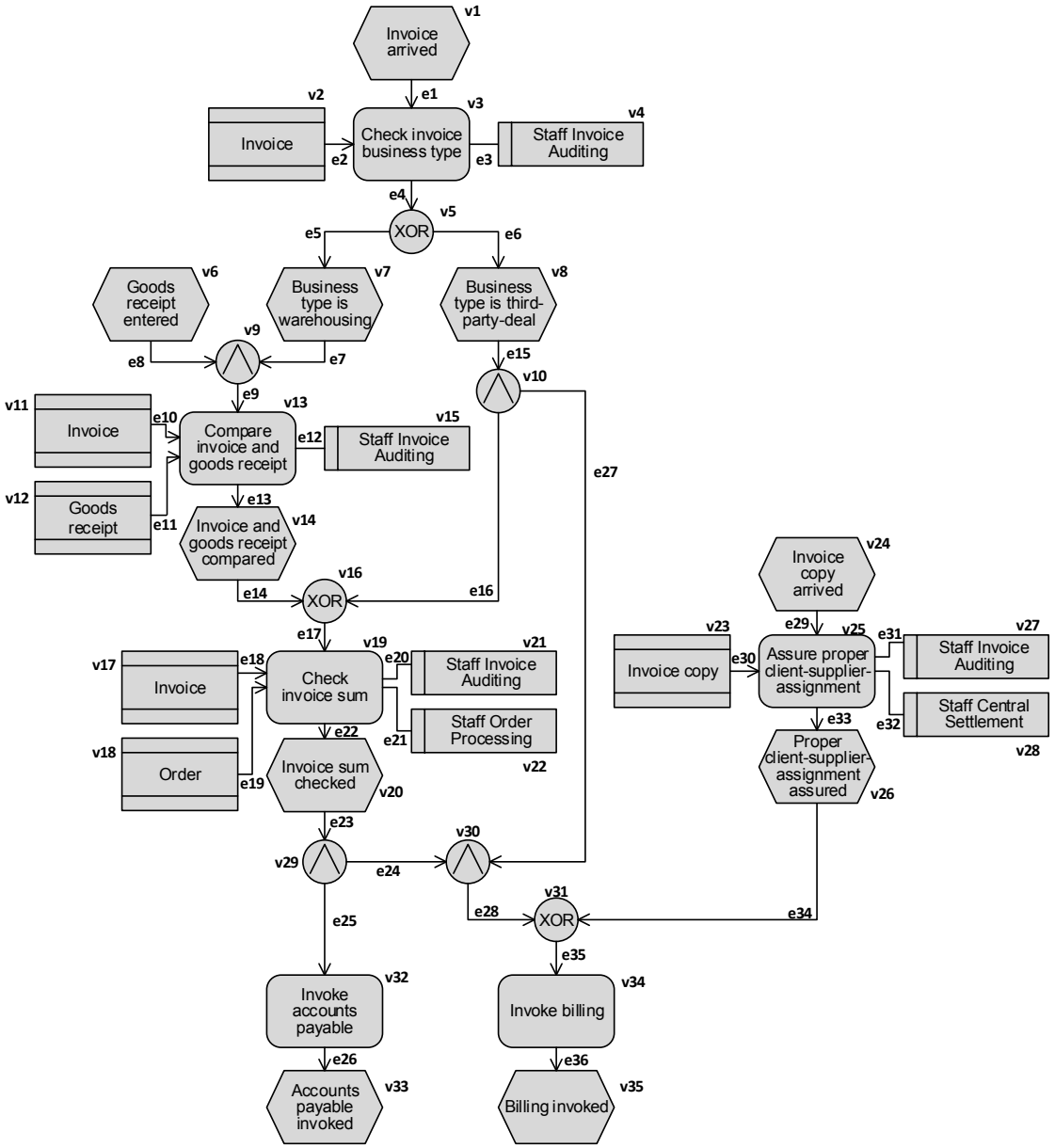
- *ControlFlows* connect *Functions*, *Events*, *XOR*, *OR*, and *AND* in any manner, except *Function*→*Function* and *Event*→*Event*.
- *Responsible* connects *Functions* with *OrgUnits*.
- *Input*: Data→Function; *Output*: Function→Data
- *directed(ControlFlow)=directed(Input)=directed(Output)=TRUE* and *directed(Responsible)=FALSE*

T: the set of all element types available; $T=T_V \cup T_E$; t is a particular element type
X_i: an arbitrary set of elements; Y_V: an arbitrary set of vertices; Y_E: an arbitrary set of edges
n_x: a natural number; *value*: a String

Set Operations	Abbreviation
ElementsOfType(X,t)	*EOT*
ElementsWithAttributeOfValue(X,t,value)	*EWAOV*
ElementsWithAttributeOfDataType(X,value)	*EWAODT*
ElementsWith{Pred|Succ}Relations(X,Y_E)	*EW{P|S}R*
ElementsWith{Pred|Succ}RelationsOfType(X,Y_E,t_E)	*EW{P|S}ROT*
ElementsWithNumberOf{Pred|Succ}Relations(X,n_x)	*EWN{P|S}R*
ElementsWithNumberOf{Pred|Succ}RelationsOfType(X,t_E,n_x)	*EWN{P|S}ROT*
ElementsDirectlyRelated(X_1,X_2)	*EDR*
AdjacentSuccessors(X_1,X_n)	*AS*
{Directed}Paths(X_1,X_n)	*{D}P*
{Directed}PathsContainingElements(X_1,X_n,X_c)	*{D}PCE*
{Directed}PathsNotContainingElements(X_1,X_n,X_c)	*{D}PNCE*
{Directed}Loops(X)	*{D}L*
{Directed}LoopsContainingElements(X,X_c)	*{D}LCE*
{Directed}LoopsNotContainingElements(X,X_c)	*{D}LNCE*

Set Operators

UNION, JOIN, INTERSECTION, INNER_INTERSECTION, COMPLEMENT, INNER_COMPLEMENT, SELF_UNION, SELF_INTERSECTION



Please provide the result sets for the following GMQL queries, when they are executed against the given EPC model (see next page). For example, the query EOT(V, XOR) would return the set {v5, v16, v31}. *Hint: please consider that some GMQL queries return* ***sets****, whereas other GMQL queries return* ***sets of sets****.*

a) EOT(V, OrgUnit) (1.5 points)

b) EWAOV(E, Caption, "Invoice") (1.5 points)

c) DP(
SELF_UNION(EWNPR(
EOT(V,XOR),
2),
EOT(V,XOR)) (3 points)

d) DP(
COMPLEMENT(
SELF_UNION(EWNPR(
EOT(V,Event),
0),
EWAOV(V,Caption,"Invoice*")
),
SELF_UNION(EWNSR(
EOT(V,Event),
0)) (7 points)

e) JOIN(
AS(
EWAOV(EOT(V,Data),Caption,"Invoice"),
EOT(V,Function)),
EDR(
EOT(V,Function),
EOT(V,OrgUnit))) (5 points)

Now think the other way round. Please provide a GMQL query, which leads to the given result set. Obviously, there are many possible solutions – you only need to specify one single correct one.

f) {v9, v10, v29, v30, e7, e8, e15, e23, e24, e27} (4 points)

g) { {v5, e6, v8, e15, v10, e27, v30, e28, v31} } (4 points)

Name: _____

Task 4: Model Query with DMQL (10 Points)

a) Build a query with DMQL that returns the same results as the GMQL query from task (3d) or, in case it is not possible, explain why. *Hint: Include all relevant specifications in your description. This also includes global rules, if necessary.* (7 points)

b) Build a query with DMQL that returns the same results as the GMQL query from task (3e) or, in case it is not possible, explain why. *Hint: Include all relevant specifications in your description. This also includes global rules, if necessary.* (3 points)

Name: _____

Task 5: Business Rules (27 Points)

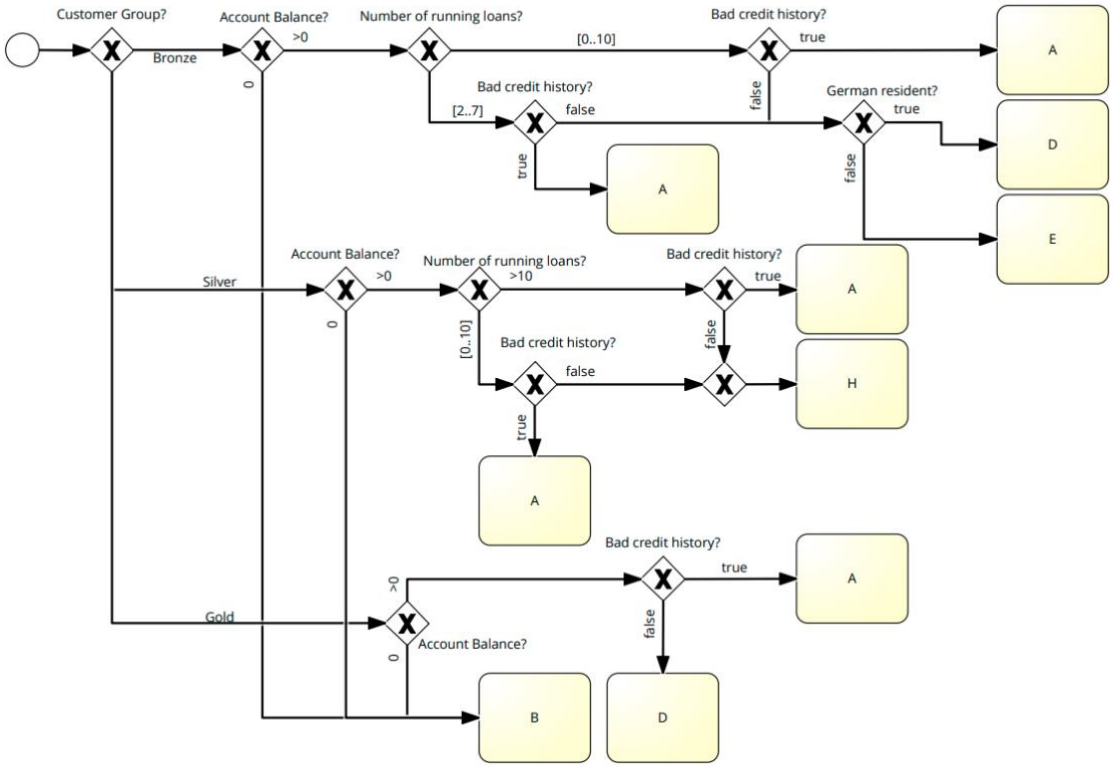
a) Consider the following DECLARE rule base. The shown rule base contains contradictions. Define the minimal contradictory subsets for the shown rule base as introduced in the lecture. *Hint: You do **not** need to draw the minimal contradictory subsets graphically.* (10 points)

ChainResponse(a,b)	*ChainResponse(e,g)*
Response(b,c)	*ChainPrecedence(c,h)*
ChainResponse(f,g)	*Response(a,c)*
Response(c,e)	*NotPrecedence(c,h)*
Response(c,d)	*ChainResponse(d,f)*
NotResponse(a,g)	*NotResponse(d,f)*
Response(d,e)	*ChainPrecedence(h,c)*

b) Which constraint(s) from Task (a) has/have the highest degree of culpability? (1 point)

c) How many distinct constraints from Task (a) are part of the contradictions? (1 point)

d) Consider the following BPMN diagram. It is an exemplary diagram for determining the risk status (a code from A-H) for a new customer loan application in a fraud management department of a financial vendor. There seems to be a lot of decision logic integrated into the model on how the risk status is determined (see the respective XOR gateways). Please abstract the shown decision logic into a new DMN Decision Table (Decision Model and Notation format). The resulting DMN Decision Table **must not** contain any errors as discussed in the lecture, i.e. there must be **no** identical rules, **no** overlapping rules and **no** subsumed rules. Your DMN Decision Table **must** contain column types and a hit indicator. Then, re-model the shown BPMN diagram by introducing a business rule task, which is annotated with the DMN decision table you created (this model should contain **no** gateways). *Hint 1: If possible, you can simplify the original (integrated) decision logic when you transform it into a DMN decision table. Hint 2: Your resulting DMN Decision Table only needs to capture the process model "as-is", i.e., you can make use of DMN Hit-Policies and rule order.* (15 points)



Name: _____

Name: _____

Good speed!

Written Examination of the Lecture
Business Process Management (BPM)
Prof. Dr. Patrick Delfmann
Summer Term 2018, 2018-07-19, 16:00h, M 001

Matriculation number: _____

Program of study: _____

Forename and surname: _____

Please note your name on **each** page.

Duration: 90 minutes
Maximum points: 90

Task		Points
Task 1: Formalization of (Process) Models	(10 Points)	
Task 2: Model Query with CTL	(25 Points)	
Task 3: Model Query with GMQL	(15 Points)	
Task 4: Model Query with DMQL	(8 Points)	
Task 5: Business Rules	(12 Points)	
Task 6: Process Mining with the α+-Algorithm	(20 Points)	
Sub-total	(90 Points)	
Bonus points from the Exercise	(+10 Points)	
Total		
Final Grade		

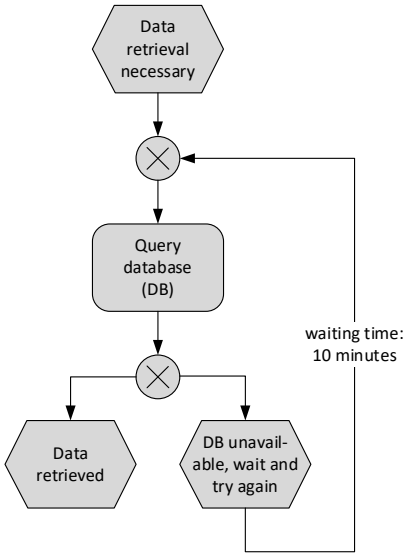
Name: _____

Task 1: Formalization of (Process) Models (10 Points)

Provide a formalization of the EPC shown on the right as introduced in the lecture. Use the following tuple as a formalization basis:

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

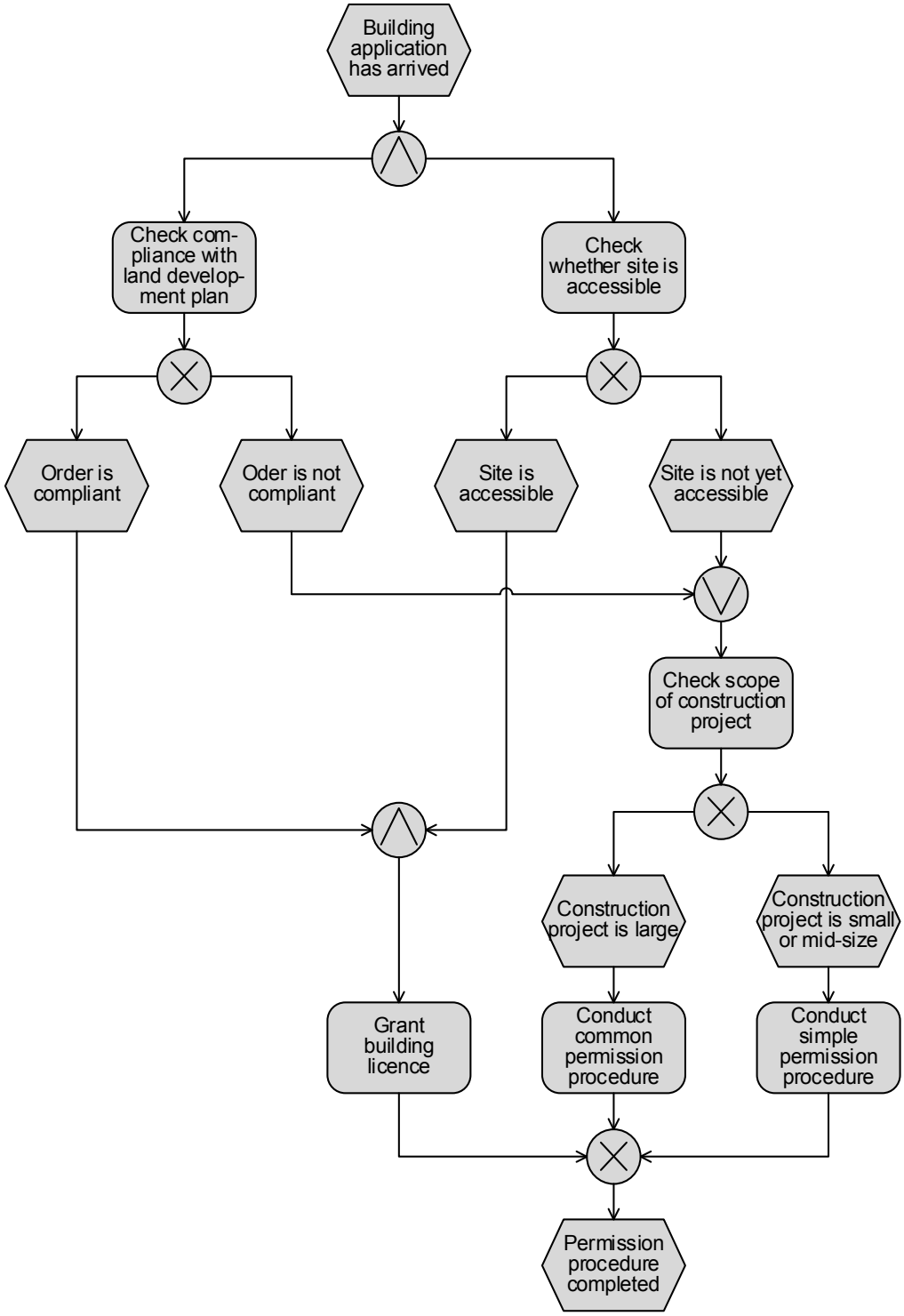
Hint: Your formalization only needs to depict this exact EPC, so you do not have to specify a full-fledged EPC language. Furthermore, you can assume that you can assign captions directly to vertices and edges of any kind.



Name: _____

Task 2: Model Query with CTL (25 Points)

The figure below shows an EPC describing the building application process of public administrations.



a) Transform the process model into a transition system, so you can examine the process with CTL. *Hint: for the transformation you only have to consider the functions, not the events.*
(10 Points)

b) Develop CTL queries that check the transition system from Task (a) for the aspects listed in subtasks (i) and (ii). Your queries should all start in the beginning state. You can use the following statements to assemble your queries φ: (15 Points)

A *("for all paths")*, ***E*** *("there exists a path")*, ***X*** *("in the next state")*, ***F*** *("in a future state")*, ***G*** *("globally in the future")*, ***U*** *("until")*

The queries φ must obey the following CTL syntax, where p is an atomic proposition:

$\varphi ::= p \mid \neg\varphi \mid \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \varphi\rightarrow\varphi \mid AX\varphi \mid EX\varphi \mid AF\varphi \mid EF\varphi \mid AG\varphi \mid EG\varphi \mid A[\varphi U\varphi] \mid E[\varphi U\varphi]$

(i) Both the development plan compliance of the building site and its accessibility have to be checked before the building permission can be granted. (5 Points)

(ii) The compliance check with the development plan and the accessibility check have to be performed concurrently (i.e., in parallel). (10 Points)

Name: _____

Task 3: Model Query with GMQL (15 Points)

Please choose from the following sets, set operations and set operators taken from the model query language GMQL introduced in the lecture to define queries for Event-driven Process Chains (EPCs). You can assume that the models to be searched for are syntactically correct. You can use the abbreviations provided in the table below when defining the queries.

Basic Sets

V: the set of all vertices available; E: the set of all edges available
Z: the set of all elements available; $Z=V \cup E$
T_V: the set of all vertex types available;
T_V={*Function, Event, XOR, OR, AND, OrgUnit, Data*}
T_E: the set of all edge types available; T_E={*ControlFlow, Responsible, Input, Output*}

- *ControlFlows* connect *Functions*, *Events*, *XOR*, *OR*, and *AND* in any manner, except *Function*→*Function* and *Event*→*Event*.
- *Responsible* connects *Functions* with *OrgUnits*.
- *Input*: Data→Function; *Output*: Function→Data
- *directed(ControlFlow)=directed(Input)=directed(Output)=TRUE* and *directed(Responsible)=FALSE*

T: the set of all element types available; $T=T_V \cup T_E$; t is a particular element type
X_i: an arbitrary set of elements; Y_V: an arbitrary set of vertices; Y_E: an arbitrary set of edges
n_x: a natural number; *value*: a String

Set Operations	Abbreviation
ElementsOfType(X,t)	*EOT*
ElementsWithAttributeOfValue(X,t,value)	*EWAOV*
ElementsWithAttributeOfDataType(X,value)	*EWAODT*
ElementsWith{Pred|Succ}Relations(X,Y_E)	*EW{P|S}R*
ElementsWith{Pred|Succ}RelationsOfType(X,Y_E,t_E)	*EW{P|S}ROT*
ElementsWithNumberOf{Pred|Succ}Relations(X,n_x)	*EWN{P|S}R*
ElementsWithNumberOf{Pred|Succ}RelationsOfType(X,t_E,n_x)	*EWN{P|S}ROT*
ElementsDirectlyRelated(X_1,X_2)	*EDR*
AdjacentSuccessors(X_1,X_n)	*AS*
{Directed}Paths(X_1,X_n)	*{D}P*
{Directed}PathsContainingElements(X_1,X_n,X_c)	*{D}PCE*
{Directed}PathsNotContainingElements(X_1,X_n,X_c)	*{D}PNCE*
{Directed}Loops(X)	*{D}L*
{Directed}LoopsContainingElements(X,X_c)	*{D}LCE*
{Directed}LoopsNotContainingElements(X,X_c)	*{D}LNCE*

Set Operators

UNION, JOIN, INTERSECTION, INNER_INTERSECTION, COMPLEMENT, INNER_COMPLEMENT, SELF_UNION, SELF_INTERSECTION

In particular, define a query for *business process weakness detection*: Situations where responsibilities of executing process activities frequently change from one person to another one are considered problematic (so-called *ping-pong responsibilities*). Define a GMQL query that finds such ping-pong responsibilities. A ping-pong responsibility is present as soon as an EPC function is executed by an orgunit, the next function is executed by another orgunit, and the function after next is again executed by the first orgunit. *Hint: please remember that in EPCs functions and events must alternate!*

Name: _____

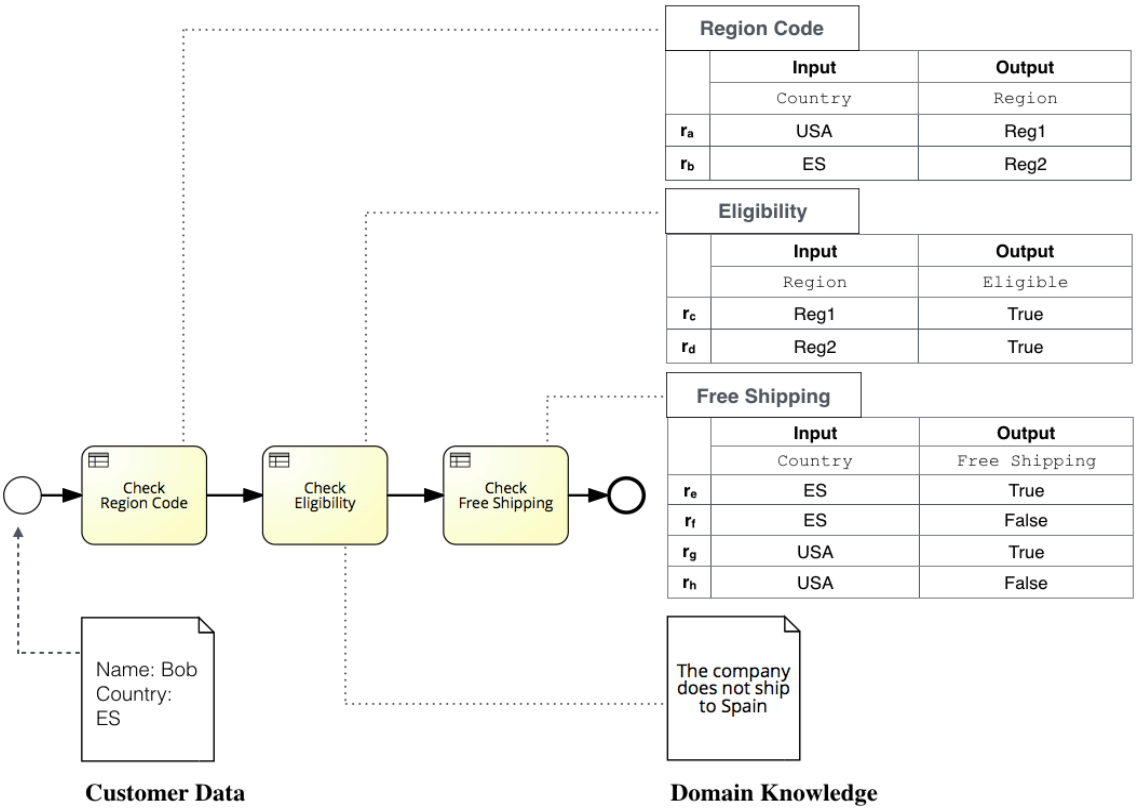
Name: _____

Task 4: Model Query with DMQL (8 Points)

Imagine the same situation as given in Task 3. Develop a corresponding DMQL query. *Hint: Consider that it might be necessary to provide some properties that are normally specified using menus, lists, etc. (e.g., edge types, path lengths, global rules or the like). Please list them using bullet points.*

Task 5: Business Rules (12 Points)

Consider the following business process. For the individual business rule tasks, the corresponding DMN tables are given. Also, external domain knowledge regarding the eligibility of customers is provided in natural language.



Region Code

	Input	Output
	Country	Region
r_a	USA	Reg1
r_b	ES	Reg2

Eligibility

	Input	Output
	Region	Eligible
r_c	Reg1	True
r_d	Reg2	True

Free Shipping

	Input	Output
	Country	Free Shipping
r_e	ES	True
r_f	ES	False
r_g	USA	True
r_h	USA	False

a) Formalize the shown *DMN tables* as introduced in the lecture. Also, formalize a rule for the *domain knowledge* regarding the eligibility of customers. Furthermore, add formalized facts as described in the *customer data*. In the following we call this set of rules and facts the knowledge base K. (5 Points)

b) Draw the minimal inconsistent subsets of the knowledge base K. (1 Point)

Name: _____

c) How many minimal inconsistent subsets are there? (1 Point)

d) How many distinct elements are part of the overall inconsistency? (1 Point)

e) Provide the $C_{\#}$ culpability value for all rules a of the knowledge base K, where $C_{\#}(K, a) = \sum_{M \in MIS(K) s.t. a \in M} \frac{1}{|M|}$ (4 Points)

Written Examination of the Lecture
Business Process Management (BPM)
PD Dr. Patrick Delfmann
Summer Term 2017, 2017-08-03, 14:00h, M001

Matriculation number: _____

Program of study: _____

Forename and surname: _____

Please note your name on **each** page.

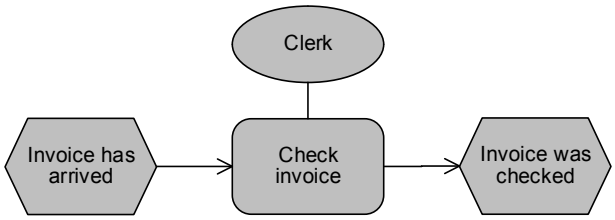
Duration: 90 minutes
Maximum points: 90

Task		Points
Task 1: Formalization of (Process) Models	(6 Points)	
Task 2: Unambiguous (Process) Models	(7 Points)	
Task 3: Model Query with CTL	(10 Points)	
Task 4: Model Query with GMQL	(21 Points)	
Task 5: Model Query with DMQL	(10 Points)	
Task 6: Process Mining with the α-Algorithm	(15 Points)	
Task 7: Process Mining with the Multi-phase Miner	(21 Points)	
Sub-total	(90 Points)	
Bonus points from the Exercise	(+10 Points)	
Total		
Final Grade		

Task 1: Formalization of (Process) Models (6 Points)

Provide a formalization of the EPC shown below as introduced in the lecture. Use the following tuple as a formalization basis:

$$M = \left(V, E, C, L, T_V, T_E, \alpha, \beta, \gamma\right)$$



Hint: Your formalization only needs to depict this exact EPC, so you do not have to specify a full-fledged EPC language.

Task 2: Unambiguous (Process) Models (7 Points)

Consider the phrase structure convention

<verb, imperative> <noun, singular, object case>

and the domain thesaurus

invoice (noun), receipt (noun), order (noun), stock (noun), check (verb), send (verb),

where "*invoice*" and "*receipt*" are synonyms and "*invoice*" is *dominant*. Further consider that a modeler enter the phrase

"bill must be verified".

Describe how the approach for unambiguous process modeling introduced in the lecture processes this phrase and what phrase(s) it actually offers to the modeler.

Task 3: Model Query with CTL (10 Points)

Please develop a CTL query that check a business process available as a transition system. Your query should start in the beginning state. You can use the following statements to assemble your query φ:

***A** ("for all paths"), **E** ("there exists a path"), **X** ("next state"), **F** ("in a future state"), **G** ("globally in the future"), **U** ("until")*

The query φ must obey the following CTL syntax, where p is an atomic proposition:

$$\varphi ::= p \mid \neg\varphi \mid \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \varphi\rightarrow\varphi \mid AX\varphi \mid EX\varphi \mid AF\varphi \mid EF\varphi \mid AG\varphi \mid EG\varphi \mid A[\varphi U\varphi] \mid E[\varphi U\varphi]$$

In business processes, responsibilities to execute activities should not change too often. Assume that we know who can execute activities in the process to be checked in principle. Further assume that these people are named using the atomic propositions *Org1* and *Org2* in the corresponding transition system. Create a query that finds out if there is some kind of ping-pong responsibility in the process. This means that the responsibility of executing activities jumps several times from one person to the other. We assume that the ping-pong responsibility becomes a problem as soon as we have *at least three responsibility **changes*** in the process. The query should return *TRUE* if we find such a ping-pong responsibility.

Task 4: Model Query with GMQL (21 Points)

Please choose from the following sets, set operations and set operators taken from the model query language GMQL introduced in the lecture to define queries for Event-driven Process Chains (EPCs). You can assume that the models to be searched for are syntactically correct. You can use the abbreviations provided in the table below when defining the queries.

Basic Sets

V: the set of all vertices available; E: the set of all edges available
Z: the set of all elements available; $Z=V \dot{\cup} E$
T_V: the set of all vertex types available;
$T_V=\{Function, Event, XOR, OR, AND, OrgUnit, Data\}$
T_E: the set of all edge types available; $T_E=\{ControlFlow, Responsible, Input, Output\}$
- *ControlFlows* connect *Functions*, *Events*, *XOR*, *OR*, and *AND* in any manner, except *Function*→*Function* and *Event*→*Event*.
- *Responsible* connects *Functions* with *OrgUnits*.
- *Input*: Data→Function; *Output*: Function→Data
- *directed(ControlFlow)=directed(Input)=directed(Output)=TRUE* and *directed(Responsible)=FALSE*

T: the set of all element types available; $T=T_V \dot{\cup} T_E$; t is a particular element type
X_i: an arbitrary set of elements
Y_V: an arbitrary set of vertices; Y_E: an arbitrary set of edges
n_x: a natural number; *value*: a String

Set Operations	Abbreviation
ElementsOfType(X,t)	*EOT*
ElementsWithAttributeOfValue(X,t,value)	*EWAOV*
ElementsWithAttributeOfDataType(X,value)	*EWAODT*
ElementsWith{Pred|Succ}Relations(X,Y_E)	*EW{P|S}R*
ElementsWith{Pred|Succ}RelationsOfType(X,Y_E,t_E)	*EW{P|S}ROT*
ElementsWithNumberOf{Pred|Succ}Relations(X,n_x)	*EWN{P|S}R*
ElementsWithNumberOf{Pred|Succ}RelationsOfType(X,t_E,n_x)	*EWN{P|S}ROT*
ElementsDirectlyRelated(X_1,X_2)	*EDR*
AdjacentSuccessors(X_1,X_n)	*AS*
{Directed}Paths(X_1,X_n)	*{D}P*
{Directed}PathsContainingElements(X_1,X_n,X_c)	*{D}PCE*
{Directed}PathsNotContainingElements(X_1,X_n,X_c)	*{D}PNCE*
{Directed}Loops(X)	*{D}L*
{Directed}LoopsContainingElements(X,X_c)	*{D}LCE*
{Directed}LoopsNotContainingElements(X,X_c)	*{D}LNCE*

Set Operators

UNION, JOIN, INTERSECTION, INNER_INTERSECTION, COMPLEMENT, INNER_COMPLEMENT, SELF_UNION, SELF_INTERSECTION

(a) Define a query for *business process weakness detection*: If we find activities in processes that contain the same name, the process might contain double work, which is considered inefficient. Build a GMQL query that finds paths, which begin and end with functions carrying the same name. (5 Points)

(b) Define another query for *business process weakness detection*: There are steps in processes where many process branches synchronize and, subsequently, split up again into many branches. Such constructs require a high amount of coordination, hence we call them *bottlenecks*. A rule of thumb says that there is a bottleneck if *at least three* process branches synchronize and split up subsequently into *at least three* other process branches. In EPCs, bottlenecks include a connector joining the incoming process branches, followed by a function, which, in turn, is followed by a connector splitting the process up again. Build a GMQL query that finds such bottlenecks. (16 Points)

Name: _____

Task 5: Model Query with DMQL (10 Points)

Imagine the same situations as given in Task 4. Develop corresponding DMQL queries. *Hint: Consider that it might be necessary to provide some properties that are normally specified using menus, lists, etc. (e.g., edge types, path lengths, global rules or the like). Please list them using bullet points.*

(a) Double work (3 Points)

(b) Bottlenecks (7 Points)

Task 4: Pattern Matching I **(15 Points)**

Please use the following sets and set operations taken from the structural pattern matching approach GMQL introduced in the lecture to define a pattern for Event-driven Process Chains (EPCs) and Organisational Charts. You can assume that the models to be searched for pattern occurrences are syntactically correct.

Basic Sets

O: the set of all objects available
R: the set of all relationships available
E: the set of all elements available; $E=O \cup R$
B_1: the set of all object types available in EPCs; B_1={*Function, Event, XOR, OR, AND, OrgUnit, Data*}
B_2: the set of all object types available in Organisational Charts; B_2={*OrgUnit*}
$B=B_1 \cup B_2$
C_1: the set of all EPC relationship types available C_1={*ControlFlow, Responsible, Support, Input, Output*};
A *ControlFlow* can be established from any object type to any object type in an EPC – except Function→Function and Event→Event.
Responsible connects Functions with OrgUnits.
directed(ControlFlow)=directed(Input)=directed(Output)=TRUE and directed(Responsible)=FALSE
C_2: the set of all Organizational Chart relationship types available; C_2={*Superordinate*}
Superordinate connects OrgUnits with OrgUnits and *directed(Superordinate)=TRUE*
$C=C_1 \cup C_2$
A: the set of all element types available; $A=B \cup C$
a: a particular element type
X_i: an arbitrary set of elements
Y: an arbitrary set of objects
Z: an arbitrary set of relationships
n_x: a natural number
value: a String

Set Functions	**Abbrev.**	**Set Operators**
ElementsOfType(X,a)	*EOT*	*UNION*
ElementsWithAttributeOfValue(X,a,value)	*EWAOV*	*JOIN*
ElementsDirectlyRelated(X_1,X_2)	*EDR*	*INTERSECTION*
ElementsWith{In|Out}Relations(X,Z)	*EW{I|O}R*	*INNER_INTERSECTION*
ElementsWith{In|Out}RelationsOfType(X,Z,c)	*EW{I|O}ROT*	*COMPLEMENT*
ElementsWithNumberOf{In|Out}Relations(X,n_x)	*EWN{I|O}R*	*INNER_COMPLEMENT*
ElementsWithNumberOf{In|Out}RelationsOfType(X,c,n_x)	*EWN{I|O}ROT*	*SELF_UNION*
DirectSuccessors(X_1,X_n)	*DS*	*SELF_INTERSECTION*
{Directed}Paths(X_1,X_n)	*{D}P*	
{Directed}PathsContainingElements(X_1,X_n,X_c)	*{D}PCE*	
{Directed}PathsNotContainingElements(X_1,X_n,X_c)	*{D}PNCE*	

Table 1: *GMQL Concepts*

Separation of duties is required in many business processes containing subsequent checks of documents, for example to ensure business process compliance. In particular, the separation of duties requires a check of a document, and later on in the process, another check of the same document. The person that conducts the second check has to be superordinate to the first one. To build an according pattern, you will have to find a corresponding structure in an EPC and, at the same time, check in an Organisational Chart whether or not the second checking person is superordinate to the first one (cf. Figure 2).

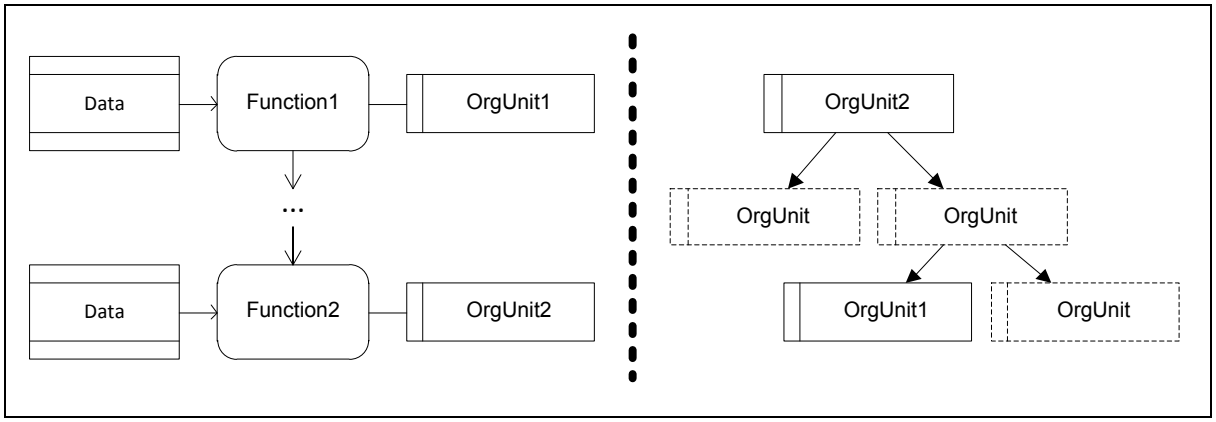


Figure 2: *Separation of Duties Pattern*

The document that is checked has to be the same one! *Hint 1: you have to build two patterns related to each other by variables. Hint 2: the following example pattern explicates the use of variables and returns a path starting and ending with the same function:*

DirectedPaths(
 ObjectsWithValues(ElementsOfType(O,Function),A),
 ObjectsWithValues(ElementsOfType(O,Function),B))≠ø
A=B

Task 4: Pattern Matching / Model Query (25 Points)

Please choose from the following sets and set operations and set operators taken from the structural pattern matching approach GMQL introduced in the lecture to define patterns for Event-driven Process Chains (EPCs). You can assume that the models to be searched for pattern occurrences are syntactically correct.

Basic Sets

O: the set of all objects available
R: the set of all relationships available
E: the set of all elements available; $E=O \cup R$
B: the set of all object types available; *B={Function, Event, XOR, OR, AND, OrgUnit, Data}*
C: the set of all relationship types available; *C={ControlFlow, Responsible, Support, Input, Output}*;
A *ControlFlow* can be established from any object type to any object type – except Function→Function and Event→Event.
Responsible connects Functions with OrgUnits.
directed(ControlFlow)=directed(Input)=directed(Output)=TRUE and *directed(Responsible)=FALSE*
A: the set of all element types available; $A=B \cup C$
a: a particular element type
X_i: an arbitrary set of elements
Y: an arbitrary set of objects
Z: an arbitrary set of relationships
n_x: a natural number
value: a String

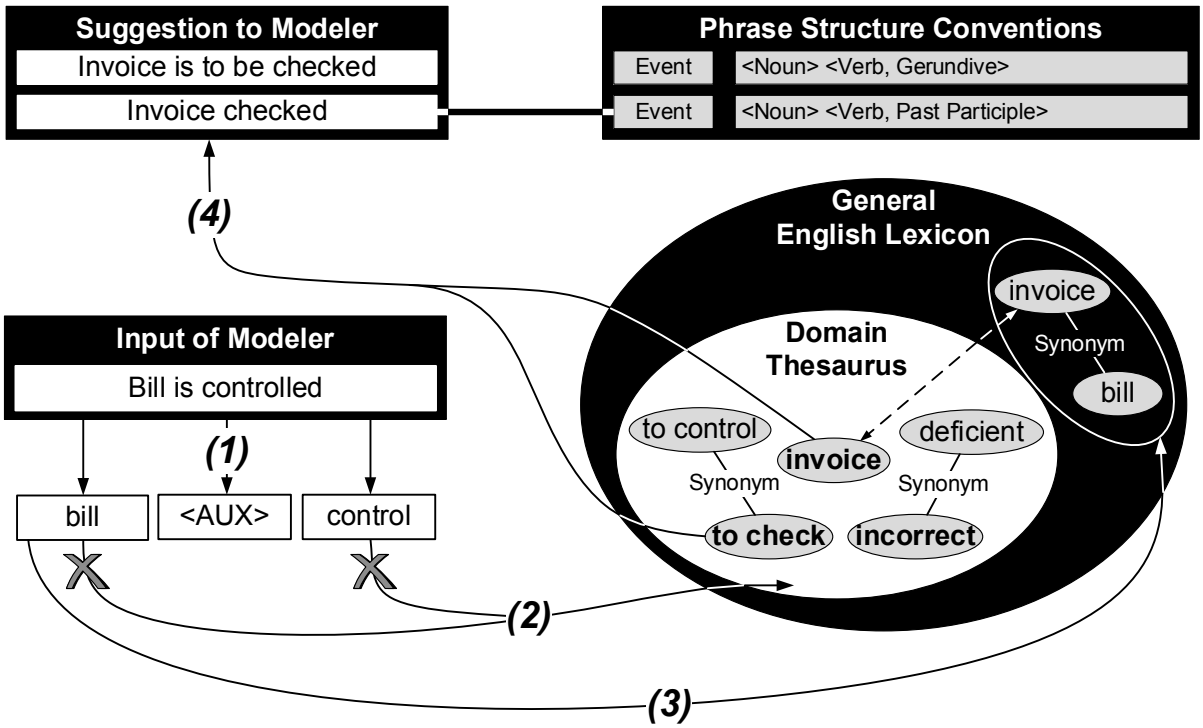
Set Operations	Abbreviation	Set Operators
ElementsOfType(X,a)	*EOT*	*UNION*
ElementsWithAttributeOfValue(X,a,value)	*EWAOV*	*JOIN*
ElementsWithAttributeOfDataType(X,a,value)	*EWAODT*	*INTERSECTION*
ElementsWith{Pred|Succ}Relations(X,Z)	*EW{P|S}R*	*INNER_INTERSECTION*
ElementsWith{Pred|Succ}RelationsOfType(X,Z,c)	*EW{P|S}ROT*	*COMPLEMENT*
ElementsWithNumberOf{Pred|Succ}Relations(X,n_x)	*EWN{P|S}R*	*INNER_COMPLEMENT*
ElementsWithNumberOf{Pred|Succ}RelationsOfType(X,c,n_x)	*EWN{P|S}ROT*	*SELF_UNION*
ElementsDirectlyRelated(X_1,X_2)	*EDR*	*SELF_INTERSECTION*
AdjacentSuccessors(X_1,X_n)	*AS*	
{Directed}Paths(X_1,X_n)	*{D}P*	
{Directed}PathsContainingElements(X_1,X_n,X_c)	*{D}PCE*	
{Directed}PathsNotContainingElements(X_1,X_n,X_c)	*{D}PNCE*	
{Directed}Loops(X_1,X_n)	*{D}L*	
{Directed}LoopsContainingElements(X_1,X_n,X_c)	*{D}LCE*	
{Directed}LoopsNotContainingElements(X_1,X_n,X_c)	*{D}LNCE*	

a) In some business processes, it is required that two specific activities ***always*** follow each other. For instance, in financial processes, it is necessary that, before a financial transaction for a customer can be executed, we have to verify her or his identity. Actually, in a related process model, it must not be possible to execute the transaction before the identity of the customer has been verified. Create an EPC pattern with GMQL that checks whether a process model ***violates*** the rule "identity check always precedes transaction". *Hints: You can simply call the activities to be searched for "transaction" and "identity check".* (14 Points)

b) In business process improvement projects, processes containing multiple changes of responsibility are regarded inefficient, as organizational barriers may impede the process flow. A rule of thumb is that in a business process, responsibility should not change more than once. Create an EPC pattern with GMQL that checks whether a process model contains more than one change of responsibility on any possible execution path. *Hint: responsibilities for the execution of process activities are expressed through annotating organizational units to activities.* (11 Points)

Task 5: Terminological Standardisation **(10 Points)**

The following figure depicts the process of terminologically standardising process model elements using an exemplary phrase input, exemplary lexicons and exemplary phrase structure conventions. The process consists of four phases (named (1)-(4) in the figure). Please describe what happens in the different phases and what technologies we could use within the phases to realize terminological standardization.



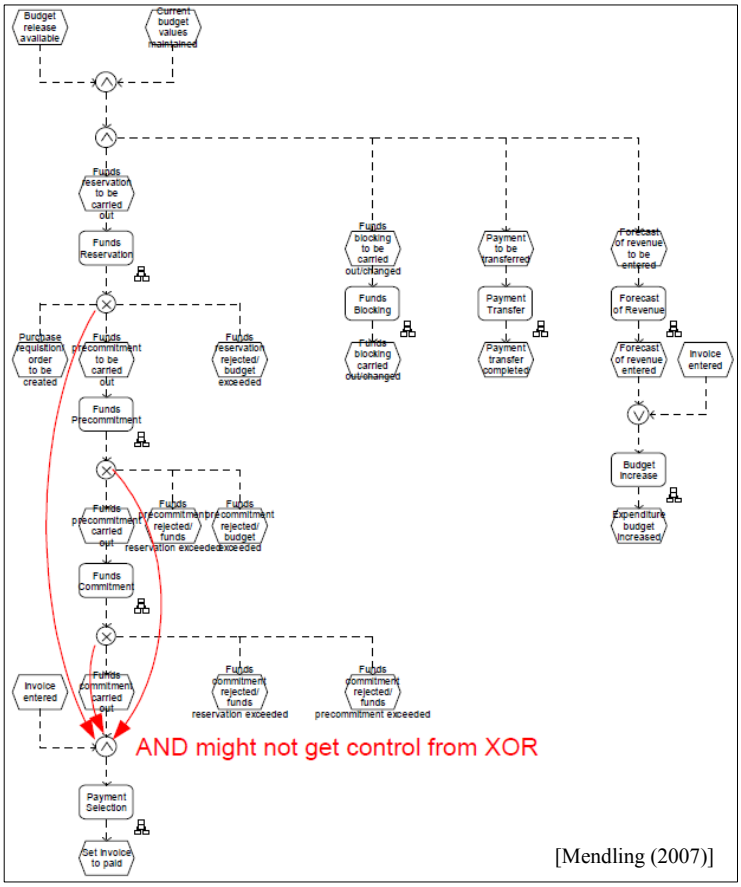
Task 6: Pattern Matching II **(15 Points)**

Please use the following sets and set operations taken from the structural pattern matching approach introduced in the lecture to define a pattern for Event-driven Process Chains (EPCs):

Basic Sets	
O: the set of all objects available; *R*: the set of all relationships available *E*: the set of all elements available; $E=O \cup R$ *B*: the set of all object types available; *B={**F**unction, **E**vent, **X**OR, **O**R, **A**ND}* *C*: the set of all relationship types available; *C={**C**ontrolFlow}*; A control flow can be established from any object type to any object type – except F→F and E→E. *c* is a particular relationship type, where *directed(c)=TRUE* $\forall c \in C$ *A*: the set of all element types available; $A=B \cup C$; *a* is a particular element type X_i: an arbitrary set of elements; *Z*: an arbitrary set of relationships n_x: a natural number	
Set Functions	**Set Operators**
ElementsOfType(X,a) *ElementsWith{In|Out}Relations(X,Z)* *ElementsWith{In|Out}RelationsOfType(X,Z,c)* *ElementsWithNumberOf{In|Out}Relations(X,n_x)* *ElementsWithNumberOf{In|Out}RelationsOfType(X,c,n_x)* *DirectSuccessorsInclRelations(X_1,X_n)*	*UNION* *JOIN* *INTERSECTION* *INNER_INTERSECTION* *COMPLEMENT* *INNER_COMPLEMENT* *SELF_UNION* *SELF_INTERSECTION*

DirectedPaths(X_1,X_n) *DirectedPathsContainingElements(X_1,X_n,X_c)* *DirectedPathsNotContainingElements(X_1,X_n,X_c)*	

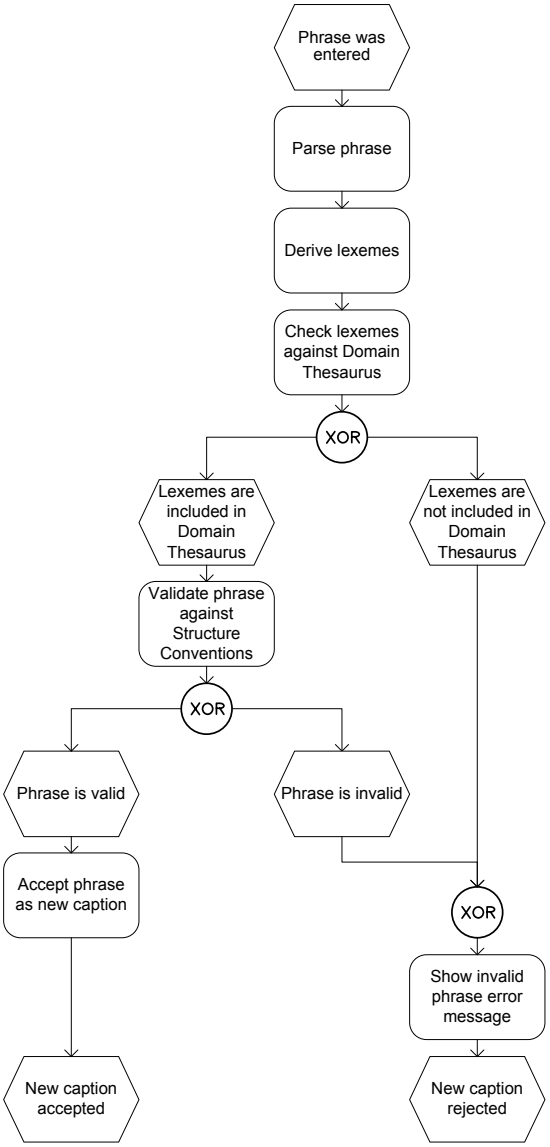
Build a pattern that describes a commonly occurring structure in an EPC process model that is no syntactical error but can cause problems in the process execution. The pattern called "AND might not get control from XOR/OR" depicts a XOR/OR split which is followed by a path of arbitrary length. At the end of that path there is a joining AND that is triggered by the path and an external event (i.e., a start event or, in other words: an event with no ingoing edge and one outgoing edge leading to the AND). In cases where the XOR/ OR split routes a process instance to another process branch and the external event occurs, the AND cannot fire and causes a situation similar to a deadlock. Such a situation is depicted in the alongside figure. Please build a pattern that considers both OR and XOR connectors!



Hints: You do not need to use every set function, set operator, and set. Just use those that are required in your opinion!

Task 7: Semantic Standardisation **(20 Points)**

The M Software Foundation has recently started a project to incorporate a semantic standardisation approach into the M^2 software. On a conceptual level a first draft has been presented to you (see figure below). Please shortly explain the pros and cons of this draft. In a next step explain how the shortcomings can be overcome.


Phrase was entered
Parse phrase
Derive lexemes
Check lexemes against Domain Thesaurus
XOR
Lexemes are included in Domain Thesaurus
Lexemes are not included in Domain Thesaurus
Validate phrase against Structure Conventions
XOR
Phrase is valid
Phrase is invalid
Accept phrase as new caption
XOR
Show invalid phrase error message
New caption accepted
New caption rejected

Task 4: Pattern Matching I **(20 Points)**

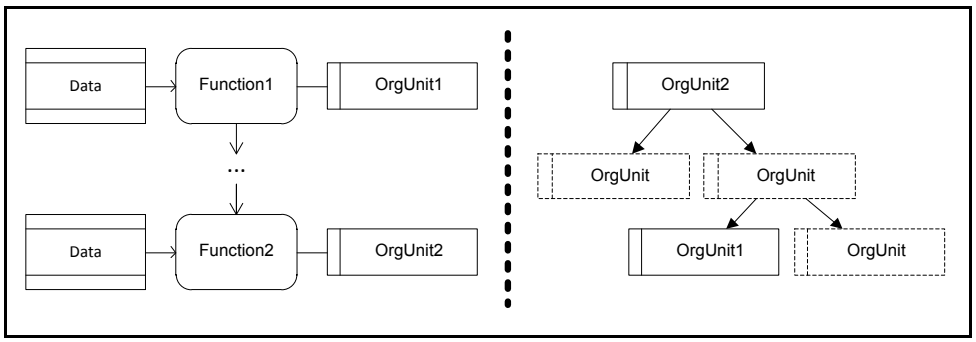
Please use the following sets and set operations taken from the structural pattern matching approach introduced in the lecture to define patterns for Event-driven Process Chains (EPCs) and Organisational Charts. You can assume that the models to be searched for pattern occurrences are syntactically correct.

Basic Sets

O: the set of all objects available
R: the set of all relationships available
E: the set of all elements available; $E=O \cup R$
B_1: the set of all object types available in EPCs; B_1={*Function, Event, XOR, OR, AND, OrgUnit, Data*}
B_2: the set of all object types available in Organisational Charts; B_2={*OrgUnit* }
$B=B_1 \cup B_2$
C_1: the set of all EPC relationship types available;
C_2={*ControlFlow, Responsible, Support, Input, Output*};
A *ControlFlow* can be established from any object type to any object type in an EPC –
except Function→Function and Event→Event.
Responsible connects Functions with OrgUnits.
directed(ControlFlow)= directed(Input)= directed(Output)=TRUE and directed(Responsible)=FALSE
C_2: the set of all Organizational Chart relationship types available; C_2={*Superordinate*}
Superordinate connects OrgUnits with OrgUnits and *directed(Superordinate)=TRUE*
$C=C_1 \cup C_2$
A: the set of all element types available; $A=B \cup C$
a: a particular element type
X_i: an arbitrary set of elements
Y: an arbitrary set of objects
Z: an arbitrary set of relationships
n_x: a natural number
value: a String

Set Functions	Abbrev.	Set Operators
ElementsOfType(X,a)	*EOT*	*UNION*
ObjectsWithValues(X,value)	*OWV*	*JOIN*
ElementsDirectlyRelatedInclRelations(X_1,X_2)	*EDR*	*INTERSECTION*
ElementsWith{In|Out}Relations(X,Z)	*EW{I|O}R*	*INNER_INTERSECTION*
ElementsWith{In|Out}RelationsOfType(X,Z,c)	*EW{I|O}ROT*	*COMPLEMENT*
ElementsWithNumberOf{In|Out}Relations(X,n_x)	*EWN{I|O}R*	*INNER_COMPLEMENT*
ElementsWithNumber-	*EWN{I|O}ROT*	*SELF_UNION*
Of{In|Out}RelationsOfType(X,c,n_x)	*DS*	*SELF_INTERSECTION*
DirectSuccessorsInclRelations(X_1,X_n)	*{D}P*	
{Directed}Paths(X_1,X_n)	*{D}PCE*	
{Directed}PathsContainingElements(X_1,X_n,X_c)	*{D}PNCE*	
{Directed}PathsNotContainingElements(X_1,X_n,X_c)		

a) The *four-eye principle* is required in many business processes containing subsequent checks of documents, for example to ensure business process compliance. In particular, the four-eye principle requires a check of a document, and later on in the process, another check of the same document. The person that conducts the second check has to be superordinate to the first one. To build an according pattern, you will have to find a corresponding structure in an EPC and, at the same time, check in an Organisational Chart whether or not the second checking person is superordinate to the first one (cf. figure below).

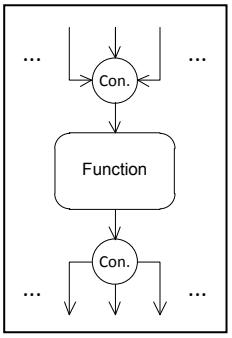


The document that is checked has to be the same one! *Hint 1: you have to build two patterns that are related to each other by variables. Hint 2: the following example pattern explicates the use of variables and returns a path starting and ending with the same function:*

```
DirectedPaths(
  ObjectsWithValues(ElementsOfType(O,Function),A),
  ObjectsWithValues(ElementsOfType(O,Function),B))≠ø
A=B
```

(12 Points)

b) Bottlenecks are weaknesses in business processes that may lead to a bad performance. Bottlenecks consist of a single function that is preceded by a multiple join and succeeded by a multiple split. Please build a pattern that identifies such bottlenecks. Assume that a bottleneck requires at least three joining process branches preceding the function and at least three process branches succeeding the function. It is of no interest what types of connectors do the split and the join. (8 Points)



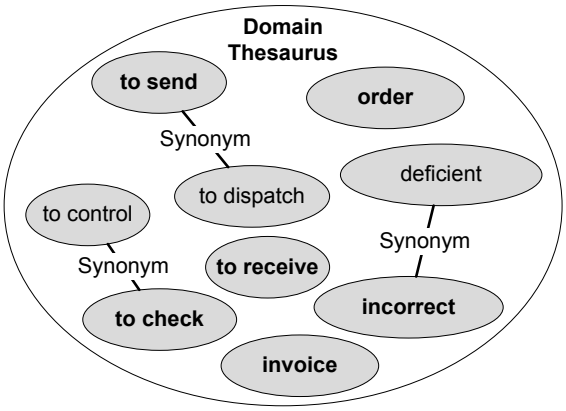
Task 6: Semantic Standardisation **(20 Points)**

a) Why is semantic unambiguity of information models a prerequisite for proper model application? Please provide an example. (5 Points)

b) Please describe all the steps necessary to semantically standardize the *function* label "Bill has to be analysed" as introduced in the lecture. Use the phrase structures and the domain vocabulary from the figure below. The dominant terms in the domain thesaurus are highlighted in bold font. As no general English lexicon is given, please build and document all

synonym relations from the words of the original label with the words in the domain thesaurus (YOU are the general lexicon in this case!). (10 Points)

Structure Conventions	
Event	<Noun> <Verb, Gerundive>
Event	<Noun> <Verb, Past Participle>
Function	<Verb, Imperative> <Noun>



c) Consider a modelling project, in which no corporate language has been defined yet. However, the models have already been developed. Consequently, they could be ambiguous. Be creative and develop an approach of creating a domain thesaurus from the existing models. Can the semantic standardization approach introduced in the lecture help here? If so, what problems would have to be taken into account? (5 Points)

Good speed!

Written Examination of the Lecture
Business Process Management (BPM)
PD Dr. Patrick Delfmann
Summer Term 2016, 2016-07-28, 16:15h

Matriculation number: _____

Program of study: _____

Prename and surname: _____

Please note your name on **each** page.

Duration: 90 minutes
Maximum points: 90

Contents

Task		Points
Task 1: BPM basics	(10 Points)	
Task 2: Model-based BPM basics	(8 Points)	
Task 3: Formalization of business processes	(12 Points)	
Task 4: Multi-perspective Process Modeling	(5 Points)	
Task 5: Model Query with GMQL	(20 Points)	
Task 6: Model Query with CTL	(20 Points)	
Task 7: Process Mining	(15 Points)	
Total		

Final grade of the examination: _____

Name: _____

Task 1: BPM basics (10 Points)

Explain the Business Process Management (BPM) Lifecycle and describe what happens in each of its phases. What is the role of process models in each phase?

Name: _____

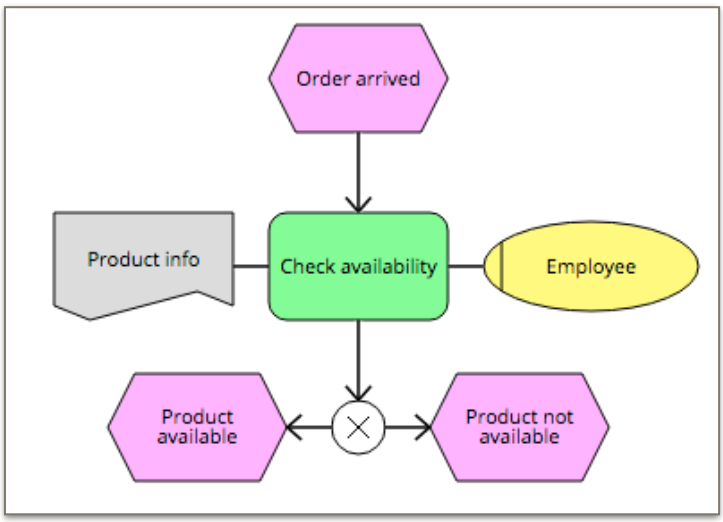
Task 2: Model-based BPM basics (8 Points)

Create a data model of a database in which you can store *conceptual* models of any kind. Hence, the database must be able to store both modeling language information and model information. Use either ER or UML Class Diagram as a modeling language to create the data model.

Name: _____

Task 3: Formalization of business processes (12 Points)

The EPC diagram depicts a process for checking the availability of goods. Please also assume the following: The function has an additional attribute „duration“, indicating the duration needed to execute the function. The duration (the attribute) is not shown in the EPC diagram, but still has to be modeled. You can assume that the shown function takes 10 minutes.



Formalize the EPC diagram as a tuple $M = (V,E,C,L,T_V,T_E,\alpha, \beta, \chi)$. The sets of M in your solution must contain all elements necessary to define the shown EPC diagram.

Name: _____

Task 5: Model Query with GMQL (20 Points)

Please choose from the following sets, set operations and set operators taken from the model query language GMQL introduced in the lecture to define queries for Event-driven Process Chains (EPCs). You can assume that the process models are terminologically standardized. When you search for vertices carrying specific names, you can use the attribute "caption" (e.g., for the input "t" of the operation EWAOV). When evaluating captions, you can use wildcards (e.g., "*invoice*" for all captions containing the word "invoice", and, eventually, some more words before and after). You can use the abbreviations provided in the table below when defining the queries.

Basic Sets

V: the set of all vertices available; E: the set of all edges available
Z: the set of all elements available; $Z=V \cup E$
T_V: the set of all vertex types available;
$T_V=\{Function, Event, XOR, OR, AND, OrgUnit, Data\}$
T_E: the set of all edge types available; $T_E=\{ControlFlow, Responsible, Input, Output\}$
- *ControlFlows* connect *Functions*, *Events*, *XOR*, *OR*, and *AND* as in EPC diagrams
- *Responsible* connects *Functions* with *OrgUnits*.
- *Input*: Data→Function; *Output*: Function→Data
- $directed(ControlFlow)=directed(Input)=directed(Output)=TRUE$ and $directed(Responsible)=FALSE$

T: the set of all element types available; $T=T_V \cup T_E$; t is a particular element type
X_i: an arbitrary set of elements
Y_V: an arbitrary set of vertices; Y_E: an arbitrary set of edges
n_x: a natural number; *value*: a String

Set Operations	Abbreviation
$ElementsOfType(X,t)$	EOT
$ElementsWithAttributeOfValue(X,t,value)$	$EWAOV$
$ElementsWithAttributeOfDataType(X,t,value)$	$EWAODT$
$ElementsWith\{Pred\|Succ\}Relations(X,Y_E)$	$EW\{P\|S\}R$
$ElementsWith\{Pred\|Succ\}RelationsOfType(X,Y_E,t_E)$	$EW\{P\|S\}ROT$
$ElementsWithNumberOf\{Pred\|Succ\}Relations(X,n_x)$	$EWN\{P\|S\}R$
$ElementsWithNumberOf\{Pred\|Succ\}RelationsOfType(X,t_E,n_x)$	$EWN\{P\|S\}ROT$
$ElementsDirectlyRelated(X_1,X_2)$	EDR
$AdjacentSuccessors(X_1,X_n)$	AS
$\{Directed\}Paths(X_1,X_n)$	$\{D\}P$
$\{Directed\}PathsContainingElements(X_1,X_n,X_c)$	$\{D\}PCE$
$\{Directed\}PathsNotContainingElements(X_1,X_n,X_c)$	$\{D\}PNCE$
$\{Directed\}Loops(X_1,X_n)$	$\{D\}L$
$\{Directed\}LoopsContainingElements(X_1,X_n,X_c)$	$\{D\}LCE$
$\{Directed\}LoopsNotContainingElements(X_1,X_n,X_c)$	$\{D\}LNCE$

Set Operators

UNION, JOIN, INTERSECTION, INNER_INTERSECTION, COMPLEMENT, INNER_COMPLEMENT, SELF_UNION, SELF_INTERSECTION

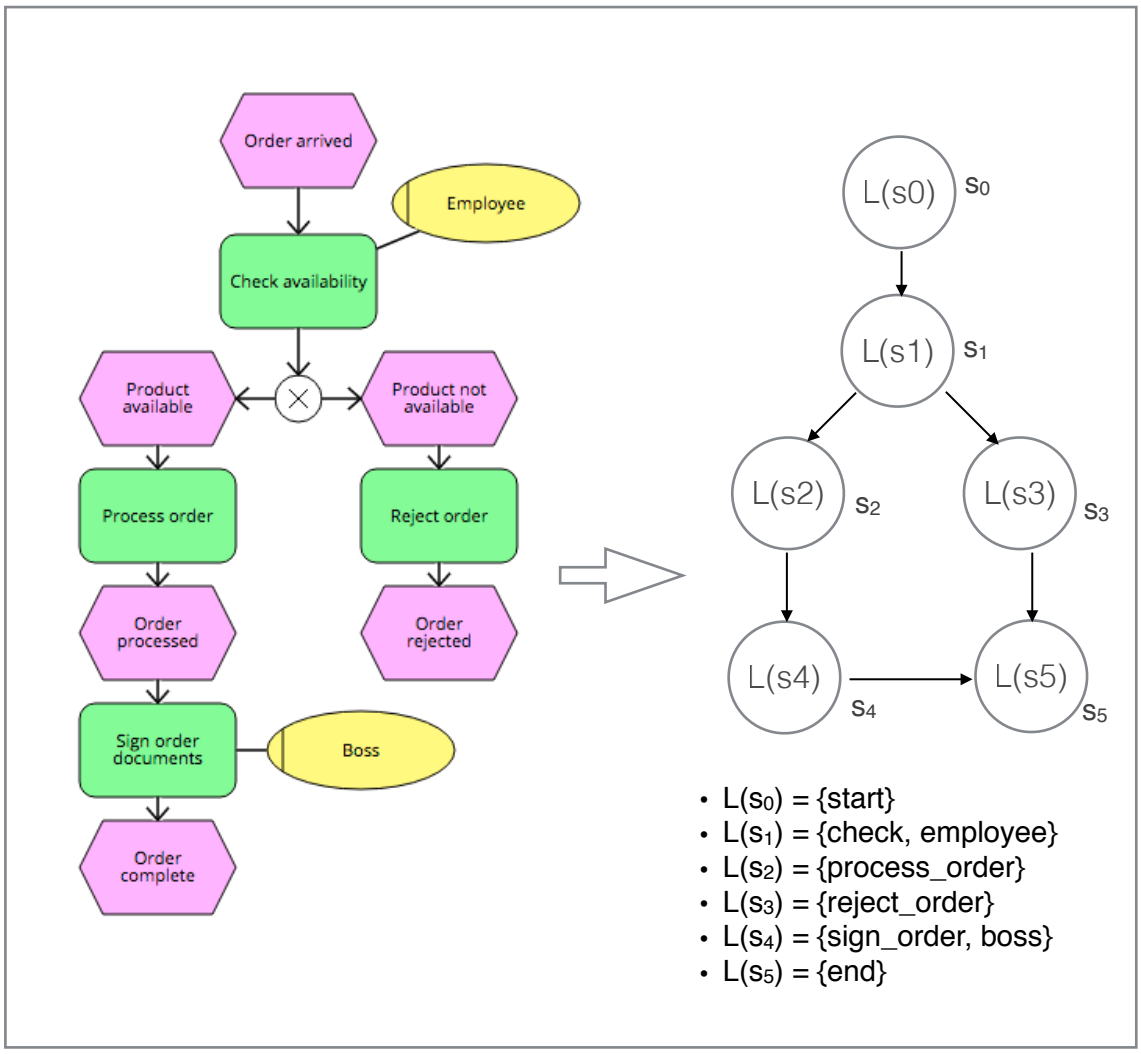
a) *Syntax checking*: In EPC diagrams, functions and events must alternate. Define a GMQL query that finds a violation of this syntax. Your query should find all cases where an event is followed by an event (without an alternating function somewhere in between) - or where a function is followed by a function (without an alternating event somewhere in between). (6 Points)

b) *Business Process Weakness Detection I*: Too many people working on the same task can be a problem for a business, as there might be communication issues. Therefore, a company might define the rule that a maximum of 2 people (=org.units) may be responsible for one task (=function). Define a GMQL query that finds violations of this rule (i.e. functions that are annotated with *more than* 2 people). (4 Points)

a) *Business Process Weakness Detection II*: If a document is printed, the same document should not be scanned again somewhere later in the process. Define a GMQL query that find violations of this process weakness (i.e. a pattern where a document is printed and somewhere later scanned). Assume that *printing* and *scanning* documents are modeled as functions and that a document is annotated to those functions. (10 Points)

(Hint: You can make some assumptions regarding the captions of the functions in the process. Please also see the hints in the introduction of this task on page 14).

Task 6: Model Query with CTL (20 Points)



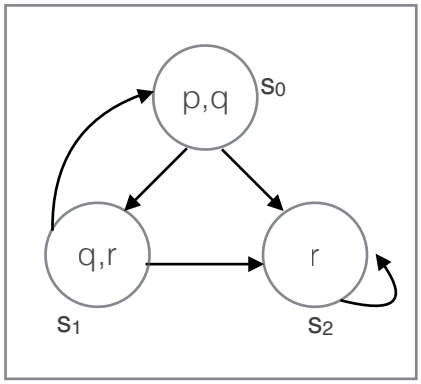
a) Depicted is an EPC diagram (left) of an order process. On the right, a corresponding transition system with its respective labels is shown. Assume a company wants to use the transition system to check for compliance with their business rules. Create CTL formulas that find violations of the business rules below: (12 Points)

- The order has to be checked, before it can be processed.
  M, s_0 |=
- Only the boss may sign the order documents.
  M, s_0 |=
- If the order is not available, the order must not be signed by the boss.
  M, s_0 |=
- The order always has to be either processed or rejected, before the process can end.
  M, s_0 |=

b) In the lecture, a transition system M was defined as a tuple $M = (S,R,L)$. S is the set of states and R is the set of relations. L is a labelling function. More specific, this labelling is defined as a function $L: S \rightarrow 2^{AP}$. *Briefly* explain this function L in your own words. What does it do and why does it say 2^{AP}? For your solution, you can assume the atomic propositions are a set AP = {A,B,C}. (3 Points)

Name: _____

c) Depicted on the right is a transition system. Explain if the following CTL formulas satisfy M in regard to the respective s ∈ S (if yes, please mark it as correct - if no, please explain why not). (5 Points)



- M, s_0 |= AF r
- M, s_0 |= EG q
- M, s_0 |= A[q U r]
- M, s_1 |= AG (p V r)
- M, s_2 |= EX (¬r)

Name: _____

Written Examination of the Lecture
Business Process Management (BPM)
PD Dr. Patrick Delfmann
Summer Term 2016, 2016-09-19, 14:15h

Matriculation number: _____

Program of study: _____

Forename and surname: _____

Please note your name on **each** page.

Duration: 90 minutes
Maximum points: 90

Contents

Task		Points
Task 1: BPM Basic Terms and Methodologies	(15 Points)	
Task 2: Formalization of (Process) Models	(20 Points)	
Task 3: Multi-perspective (Process) Models	(10 Points)	
Task 4: Model Query with GMQL	(15 Points)	
Task 5: Model Query with CTL	(20 Points)	
Task 6: Process Mining	(10 Points)	
Total		

Final grade of the examination: _____

Task 1: BPM Basic Terms and Methodologies (15 Points)

In the lecture, we introduced two different BPM methodologies, namely the Business Process Management Lifecycle by WESKE and the Process Management Methodology by BECKER, KUGELER and ROSEMANN. Briefly describe the different phases of each methodology. Further describe what phases of the one methodology resemble those of the other one and why. Explain the commonalities and differences.

Name: _____

Name: _____

Task 2: Formalization of (Process) Models (20 Points)

Please provide a generalized formalization of both conceptual models and modeling languages using set notation. Also provide a formalization of each component of models and modeling languages. Use a particular model of your choice to exemplarily “fill” the sets of your formalization.

Name: _____

Name: _____

Task 4: Model Query with GMQL (15 Points)

Please choose from the following sets, set operations and set operators taken from the model query language GMQL introduced in the lecture to define queries for Event-driven Process Chains (EPCs). You can assume that the models to be searched for are syntactically correct. You can use the abbreviations provided in the table below when defining the queries.

Basic Sets

V: the set of all vertices available; E: the set of all edges available
Z: the set of all elements available; $Z=V \dot{\cup} E$
T_V: the set of all vertex types available;
$T_V=\{Function, Event, XOR, OR, AND, OrgUnit, Data\}$
T_E: the set of all edge types available; $T_E=\{ControlFlow, Responsible, Input, Output\}$
- *ControlFlows* connect *Functions*, *Events*, *XOR*, *OR*, and *AND* in any manner, except *Function*?*Function* and *Event*?*Event*.
- *Responsible* connects *Functions* with *OrgUnits*.
- *Input*: Data?Function; *Output*: Function?Data
- *directed(ControlFlow)=directed(Input)=directed(Output)=TRUE* and *directed(Responsible)=FALSE*

T: the set of all element types available; $T=T_V \dot{\cup} T_E$; t is a particular element type
X_i: an arbitrary set of elements
Y_V: an arbitrary set of vertices; Y_E: an arbitrary set of edges
n_x: a natural number; *value*: a String

Set Operations	Abbreviation
ElementsOfType(X,t)	*EOT*
ElementsWithAttributeOfValue(X,t,value)	*EWAOV*
ElementsWithAttributeOfDataType(X,t,value)	*EWAODT*
ElementsWith{Pred|Succ}Relations(X,Y_E)	*EW{P|S}R*
ElementsWith{Pred|Succ}RelationsOfType(X,Y_E,t_E)	*EW{P|S}ROT*
ElementsWithNumberOf{Pred|Succ}Relations(X,n_x)	*EWN{P|S}R*
ElementsWithNumberOf{Pred|Succ}RelationsOfType(X,t_E,n_x)	*EWN{P|S}ROT*
ElementsDirectlyRelated(X_1,X_2)	*EDR*
AdjacentSuccessors(X_1,X_n)	*AS*
{Directed}Paths(X_1,X_n)	*{D}P*
{Directed}PathsContainingElements(X_1,X_n,X_c)	*{D}PCE*
{Directed}PathsNotContainingElements(X_1,X_n,X_c)	*{D}PNCE*
{Directed}Loops(X_1,X_n)	*{D}L*
{Directed}LoopsContainingElements(X_1,X_n,X_c)	*{D}LCE*
{Directed}LoopsNotContainingElements(X_1,X_n,X_c)	*{D}LNCE*

Set Operators

UNION, JOIN, INTERSECTION, INNER_INTERSECTION, COMPLEMENT, INNER_COMPLEMENT, SELF_UNION, SELF_INTERSECTION

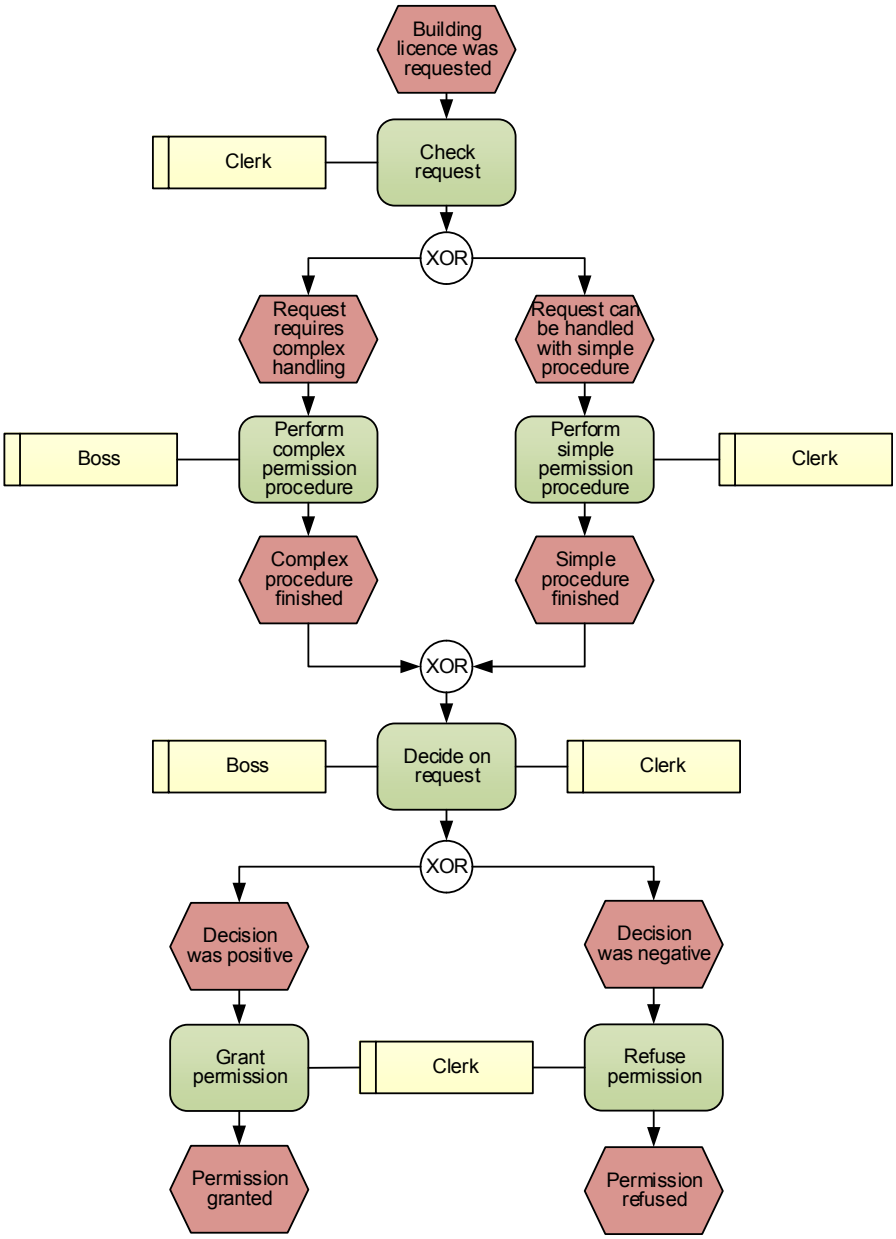
a) *Semantics Check*: A connector loop (i.e., a loop containing connectors [AND, OR, XOR] only) in an EPC is syntactically correct, however it makes no sense and confuses the model user. Specify a query that finds such connector loops. (5 Points)

b) *Business Process Compliance Checking*: A common compliance rule is the following: Before a customer can open a bank account, the bank must determine the correct identity of the customer. Define a query that finds violations of this rule, that is, where an account can be opened without previously determining the identity of the customer. (10 Points)

Name: _____

Task 6: Model Query with CTL (20 Points)

a) Consider the process model shown below. It represents a building permission process in public administration. Please transform the process model into a transition system so that you can analyze it with CTL. (9 Points)



Name: _____

b) Please develop CTL queries that check the transition system from Task (a) for the aspects listed in Tasks (i) to (iii). Your queries should all start in the beginning state. You can use the following statements to assemble your queries **ϕ**: (11 Points)

***A** ("for all paths"), **E** ("there exists a path"), **X** ("next state"), **F** ("in a future state"), **G** ("globally in the future"), **U** ("until")*

The queries **ϕ** must obey the following CTL syntax, where **p** is an atomic proposition:

$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid AX\phi \mid EX\phi \mid AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid A[\phi U \phi] \mid E[\phi U \phi]$

(i) Due to efficiency considerations in public administration, it must be possible to execute the building permission process without interaction of the boss (i.e., there must be at least one execution path that can be handled exclusively by the clerk). This way, public administration can avoid idle times effectively. The query should return "true" if there is such a path. (4 Points)

(ii) Only the boss can perform a complex permission procedure. If the query finds a place in the transition system, where someone else performs a complex permission procedure, it should return "false". (2 Points)

(iii) If the complex permission procedure was performed, then the boss must decide on the request. If the simple one was performed, the clerk must decide on the request. The query should return "true", if the transition system complies with this rule. (5 Points)