Query Stability in Data-aware Business Processes*

Ognjen Savković

unibz

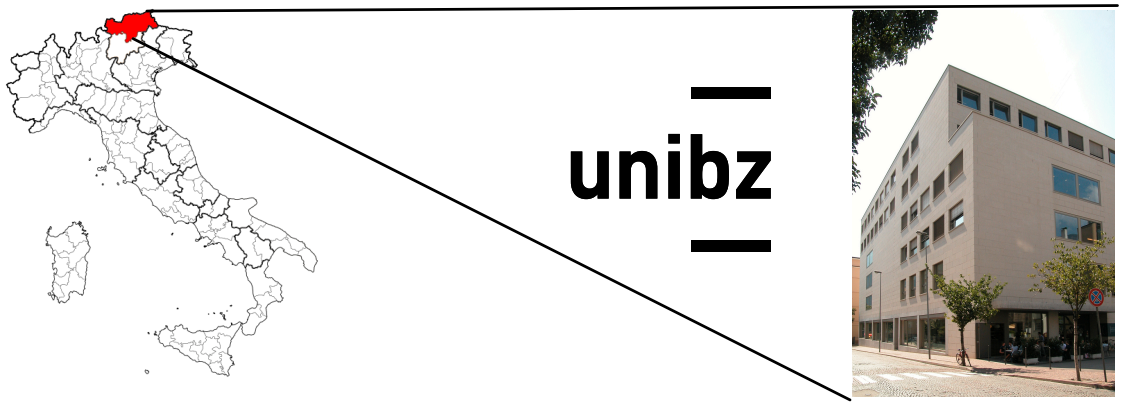
Free University of Bozen-Bolzano

joint work with
Elisa Marengo and **Werner Nutt**

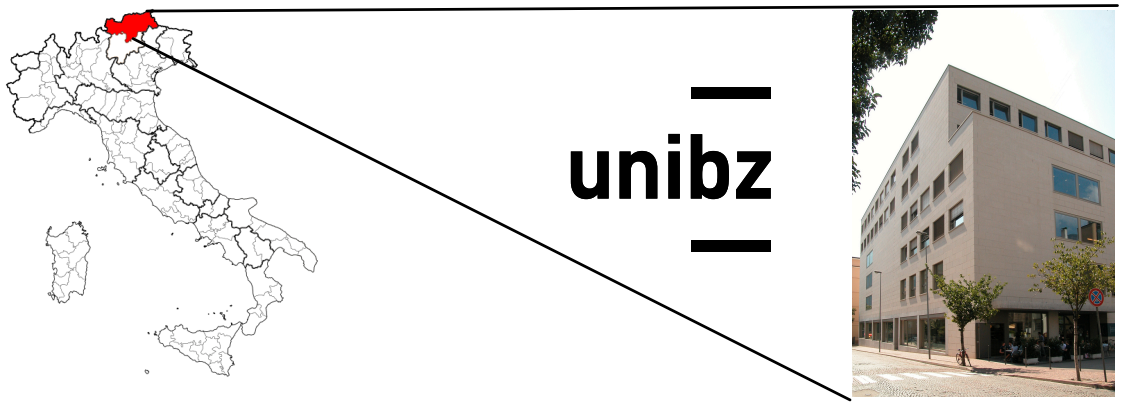
EPCL PhD Workshop, April 2014, Dresden

*Supported by the project MAGIC, funded by the Province of Bozen-Bolzano

Students at Free University of Bozen-Bolzano (FUB)

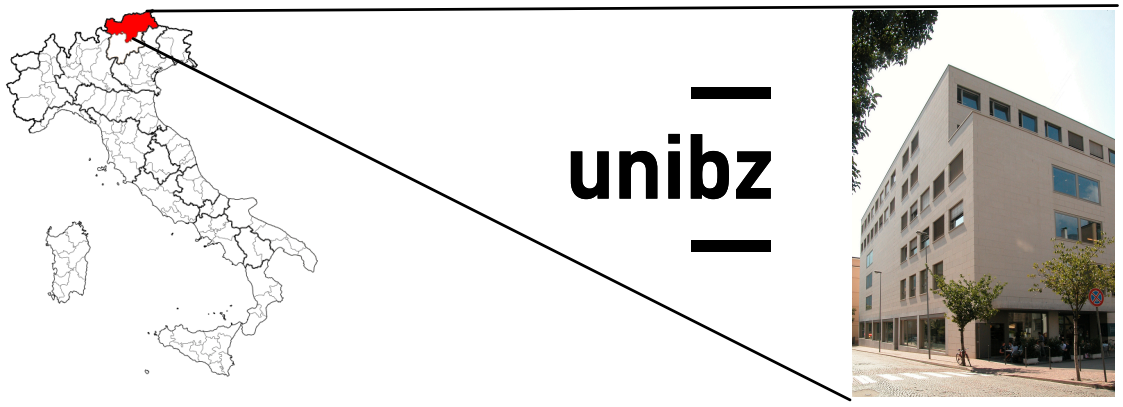


Students at Free University of Bozen-Bolzano (FUB)



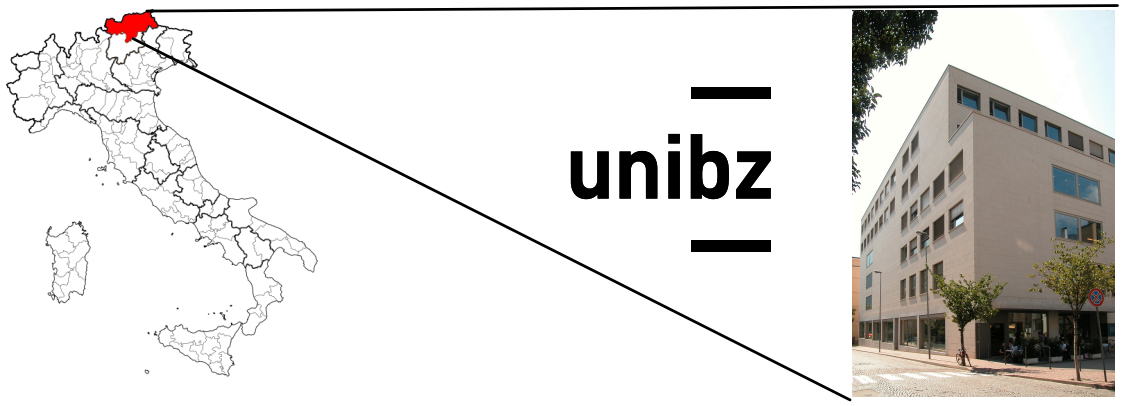
- FUB has around 3,500 students

Students at Free University of Bozen-Bolzano (FUB)



- FUB has around 3,500 students
- Dean of the Faculty of Computer Science: “Every student is a precious flower”

Students at Free University of Bozen-Bolzano (FUB)



- FUB has around 3,500 students
- Dean of the Faculty of Computer Science: “Every student is a precious flower”

Statistical Report about the Enrolled Students at FUB

Faculty of Economics	Student places 2012 / 2013	Enrollments 2011 / 2012	Enrollments 2012 / 2013	%
Bachelor in Economics and Management	[illegible]	[illegible]	[illegible]	20.78%
Bachelor in Tourism, Sport and Event Management	[illegible]	[illegible]	[illegible]	74.51%
Bachelor in Economics and Social Sciences	[illegible]	[illegible]	[illegible]	17.24%
Master in Entrepreneurship and Innovation	[illegible]	[illegible]	[illegible]	114.29%
Master in Economics and Management of the public sector	[illegible]	[illegible]	[illegible]	22.73%
Sum	[illegible]	[illegible]	[illegible]	41.45%

Faculty of Computer Science	Student Places 2012 / 2013	Enrollments 2011 / 2012	Enrollments 2012 / 2013	%
Bachelor in Computer Science and Engineering	[illegible]	[illegible]	[illegible]	30.43%
Master of Science in Computer Science	[illegible]	[illegible]	[illegible]	-44.44%
PhD in Computer Science	[illegible]	[illegible]	[illegible]	#VALUE!
Sum	[illegible]	[illegible]	[illegible]	-25.37%

How Reliable are the Figures in the Report?

Faculty of Economics	Student places 2012 / 2013	Enrollments 2011 / 2012	Enrollments 2012 / 2013	%
Bachelor in Economics and Management	[illegible]	77	93	20.78%
Bachelor in Tourism, Sport and Event Management	[illegible]	51	89	74.51%
Bachelor in Economics and Social Sciences	[illegible]	29	34	17.24%
Master in Entrepreneurship and Innovation	[illegible]	14	30	114.29%
Master in Economics and Management of the public sector	[illegible]	[illegible]	[illegible]	22.73%
Sum	176	181	272	41.45%

Faculty of Computer Science	Student Places 2012 / 2013	Enrollments 2011 / 2012	Enrollments 2012 / 2013	%
Bachelor in Computer Science and Engineering	[illegible]	69	90	30.43%
Master of Science in Computer Science	[illegible]	36	20	-44.44%
PhD in Computer Science	[illegible]	2	[illegible]	#VALUE!
Sum	135	67	50	-25.37%

- How **reliable (stable)** are the figures that we see?

How Reliable are the Figures in the Report?

Faculty of Economics	Student places 2012 / 2013	Enrollments 2011 / 2012	Enrollments 2012 / 2013	%
Bachelor in Economics and Management	[illegible]	72	87	20.78%
Bachelor in Tourism, Sport and Event Management	[illegible]	51	89	74.51%
Bachelor in Economics and Social Sciences	[illegible]	29	34	17.24%
Master in Entrepreneurship and Innovation	[illegible]	14	30	114.29%
Master in Economics and Management of the public sector	[illegible]	[illegible]	[illegible]	22.73%
Sum	176	181	272	41.45%

Faculty of Computer Science	Student Places 2012 / 2013	Enrollments 2011 / 2012	Enrollments 2012 / 2013	%
Bachelor in Computer Science and Engineering	[illegible]	92	120	30.43%
Master of Science in Computer Science	[illegible]	36	20	-44.44%
PhD in Computer Science	[illegible]	9	[illegible]	#VALUE!
Sum	135	67	50	-25.37%

- How **reliable (stable)** are the figures that we see?
- What are the **main factors** that determine how the **data** may **change** in the future?

How Reliable are the Figures in the Report?

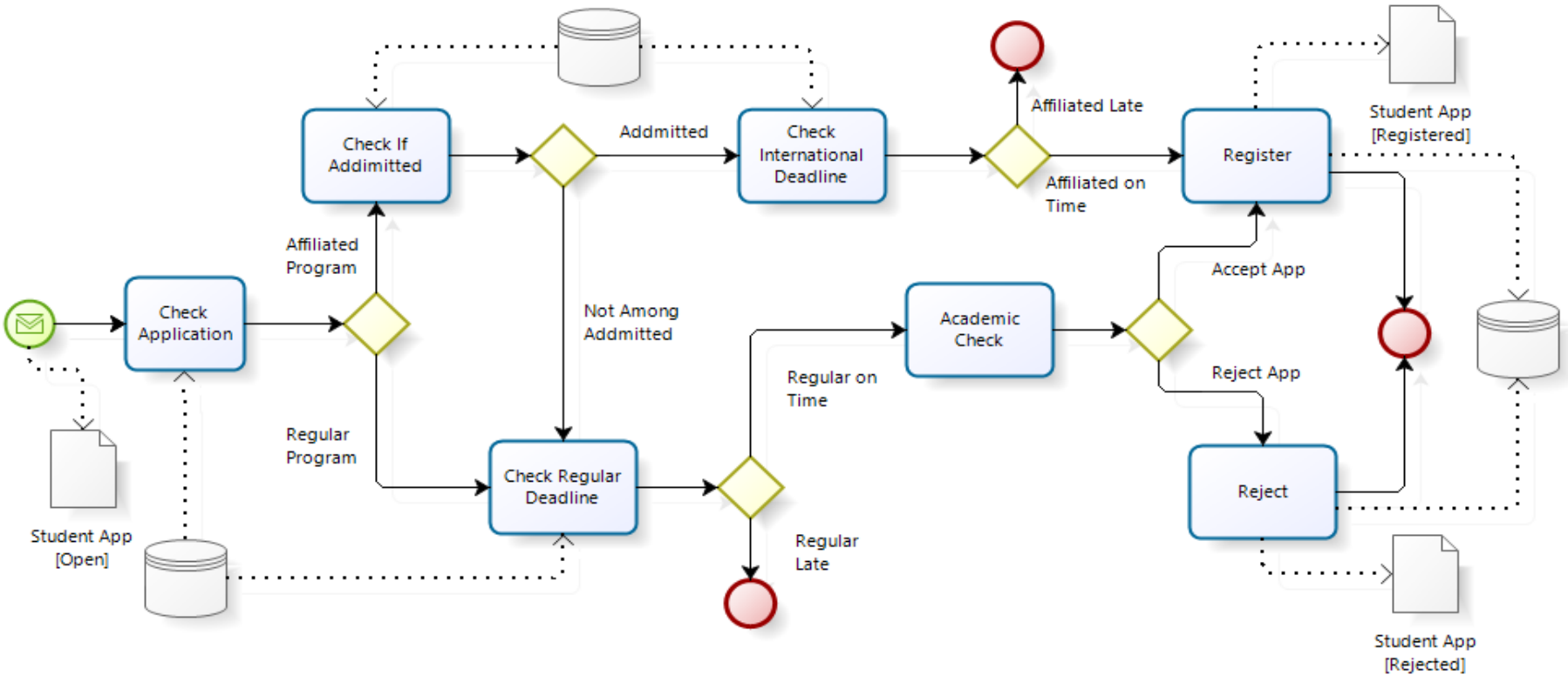
Faculty of Economics	Student places 2012 / 2013	Enrollments 2011 / 2012	Enrollments 2012 / 2013	%
Bachelor in Economics and Management	[illegible]	72	87	20.78%
Bachelor in Tourism, Sport and Event Management	[illegible]	57	99	74.51%
Bachelor in Economics and Social Sciences	[illegible]	29	34	17.24%
Master in Entrepreneurship and Innovation	[illegible]	14	30	114.29%
Master in Economics and Management of the public sector	[illegible]	[illegible]	[illegible]	22.73%
Sum	176	181	272	41.45%

Faculty of Computer Science	Student Places 2012 / 2013	Enrollments 2011 / 2012	Enrollments 2012 / 2013	%
Bachelor in Computer Science and Engineering	[illegible]	69	90	30.43%
Master of Science in Computer Science	[illegible]	36	20	-44.44%
PhD in Computer Science	[illegible]	2	[illegible]	#VALUE!
Sum	135	67	50	-25.37%

- How **reliable (stable)** are the figures that we see?
- What are the **main factors** that determine how the **data** may **change** in the future?

Look at the **Business Processes** that **generates** and **manipulates data**.

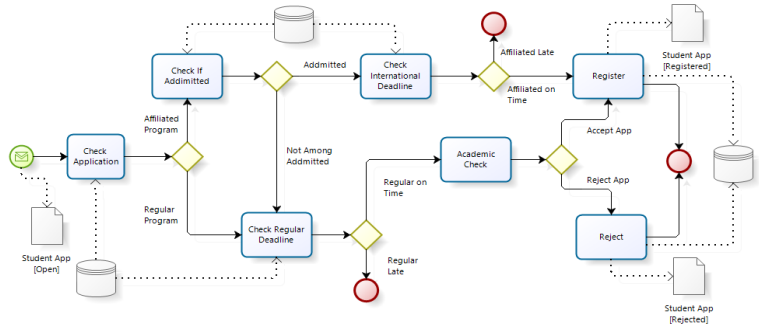
Student Registration Processes at FUB



What are Business Processes (BPs)?

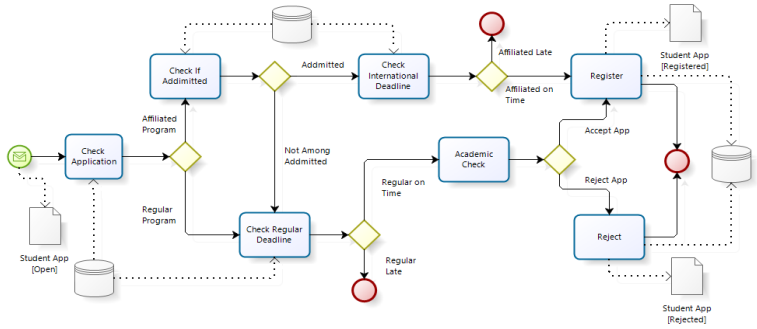
What are Business Processes (BPs)?

- BPs are sequence of **connected activities** organized to accomplish certain **goal**
  - e.g., student registration process



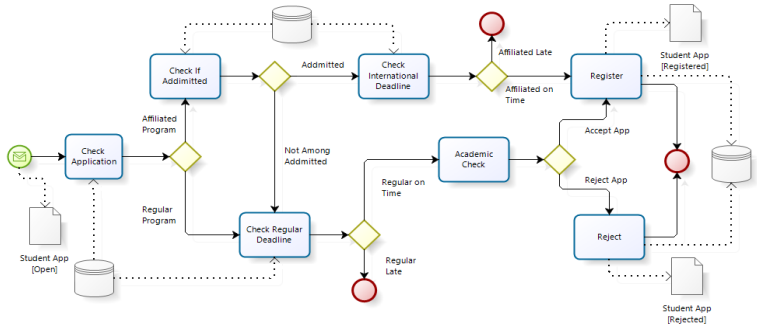
What are Business Processes (BPs)?

- BPs are sequence of **connected activities** organized to accomplish certain **goal**
  - e.g., student registration process
- Several **standardized languages**
  - e.g., `BPMN`, `BPEL`



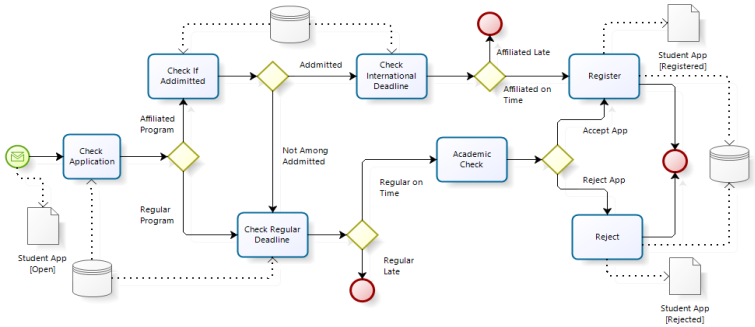
What are Business Processes (BPs)?

- BPs are sequence of **connected activities** organized to accomplish certain **goal**
  - e.g., student registration process
- Several **standardized languages**
  - e.g., `BPMN`, `BPEL`
- Exist **execution engines** that executes them
  - e.g., `jBMN`



What are Business Processes (BPs)?

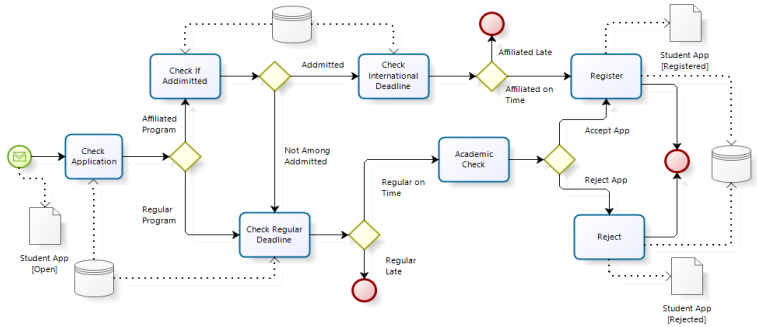
- BPs are sequence of **connected activities** organized to accomplish certain **goal**
  - e.g., student registration process
- Several **standardized languages**
  - e.g., `BPMN`, `BPEL`
- Exist **execution engines** that executes them
  - e.g., `jBMN`



 What BPs **fail to represent?**

What are Business Processes (BPs)?

- BPs are sequence of **connected activities** organized to accomplish certain **goal**
  - e.g., student registration process
- Several **standardized languages**
  - e.g., `BPMN`, `BPEL`
- Exist **execution engines** that executes them
  - e.g., `jBMN`

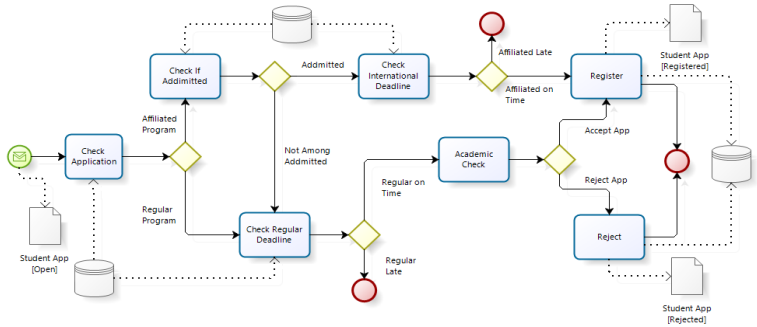


What BPs **fail to represent?**

- BPs **fail to model** interaction with **databases**
  - Formal BPs models, e.g. *Petri Nets*, traditionally **represent data in a limited way**
  - In `BPEL` operations on the database are **hidden in the code**

What are Business Processes (BPs)?

- BPs are sequence of **connected activities** organized to accomplish certain **goal**
  - e.g., student registration process
- Several **standardized languages**
  - e.g., `BPMN`, `BPEL`
- Exist **execution engines** that executes them
  - e.g., `jBMN`



What BPs **fail to represent?**

- BPs **fail to model** interaction with **databases**
  - Formal BPs models, e.g. *Petri Nets*, traditionally **represent data in a limited way**
  - In `BPEL` operations on the database are **hidden in the code**
- However, **data** is often the **main driver** when **executing** BPs
  - E.g., a student can register for a program
    only if the student was firstly admitted to the program

Property of Query Stability

Query Stability

Informally, that is when for a given query Q and a business process $\mathscr{B}$ that manipulates data the **query answer** of Q **does not change** for all future transformations of data according to $\mathscr{B}$.

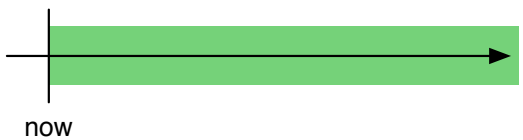
Property of Query Stability

Query Stability

Informally, that is when for a given query Q and a business process $\mathscr{B}$ that manipulates data the **query answer** of Q **does not change** for all future transformations of data according to $\mathscr{B}$.

We would like to answer the following questions

- Is query Q **stable** (from now)?



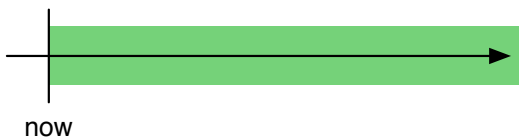
Property of Query Stability

Query Stability

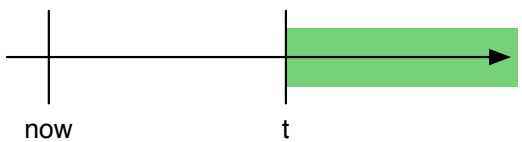
Informally, that is when for a given query Q and a business process $\mathscr{B}$ that manipulates data the **query answer** of Q **does not change** for all future transformations of data according to $\mathscr{B}$.

We would like to answer the following questions

- Is query Q **stable** (from now)?



- If not, is there a **time point** from which Q becomes stable?



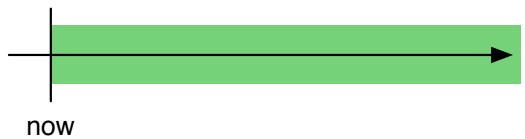
Property of Query Stability

Query Stability

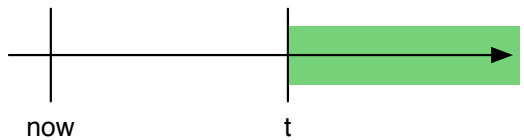
Informally, that is when for a given query Q and a business process $\mathscr{B}$ that manipulates data the **query answer** of Q **does not change** for all future transformations of data according to $\mathscr{B}$.

We would like to answer the following questions

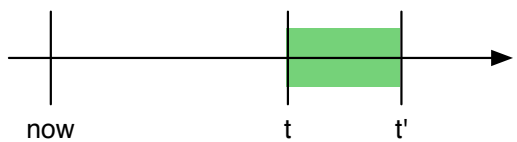
- Is query Q **stable** (from now)?



- If not, is there a **time point** from which Q becomes stable?



- What are the **time intervals** in which Q is stable?



Outline

Data-aware Business Processes (DABPs) model

Query Stability

Reasoning about Query Stability in DABPs

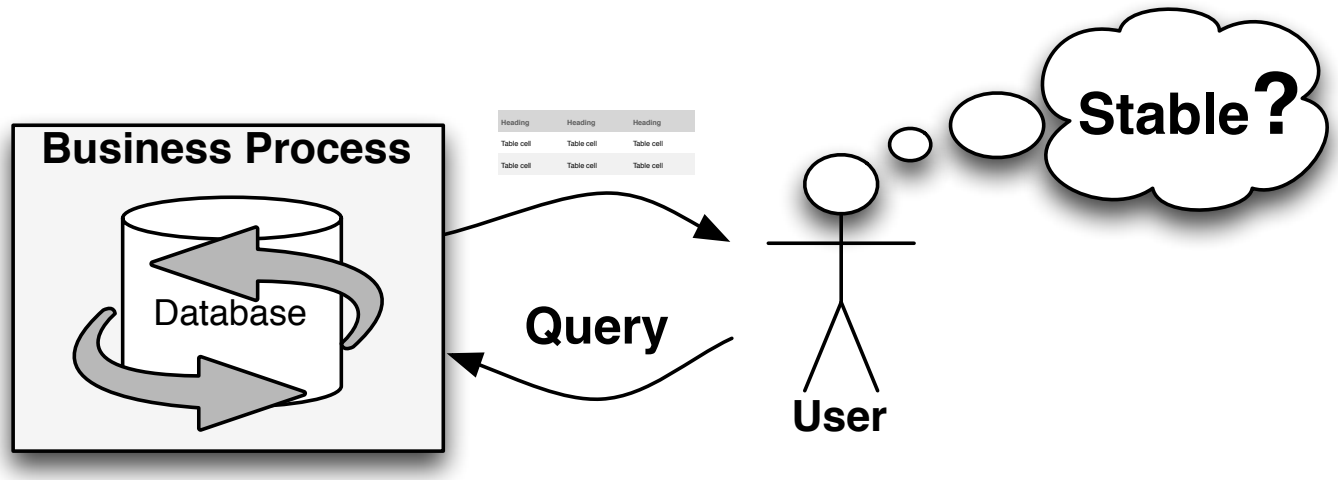


Table of Contents

Data-aware Business Processes (DABPs) model

- A **DABP** $\mathscr{B}$ consists of two parts:
  a static **Process Part** $\mathscr{P}$ and a dynamic **Data Part** $\mathscr{C}$

Data-aware Business Processes (DABPs) model

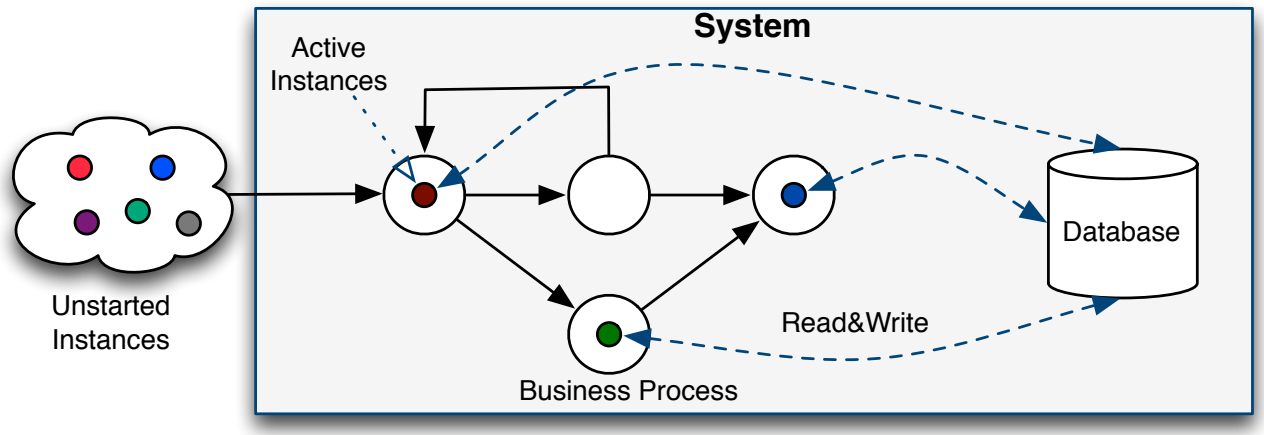
- A **DABP** $\mathscr{B}$ consists of two parts:
  a static **Process Part** $\mathscr{P}$ and a dynamic **Data Part** $\mathscr{C}$
- **Process Part** describes how the data from the data part is **read and written**

Data-aware Business Processes (DABPs) model

- A **DABP** $\mathscr{B}$ consists of two parts:
  a static **Process Part** $\mathscr{P}$ and a dynamic **Data Part** $\mathscr{C}$
- **Process Part** describes how the data from the data part is **read and written**
- **Data part** contains **database instance** and
  the set of currently **active process instances**

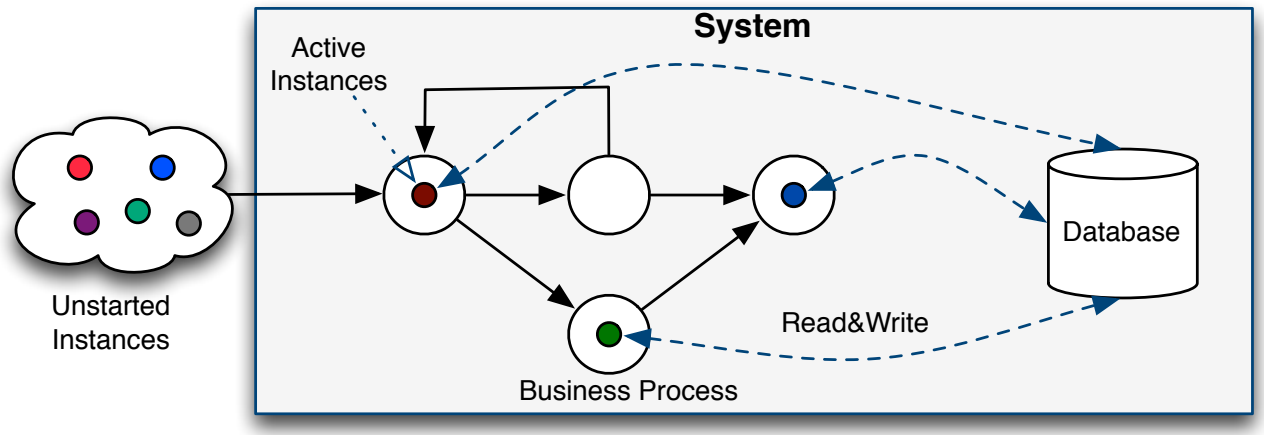
Data-aware Business Processes (DABPs) model

- A **DABP** $\mathscr{B}$ consists of two parts:
  a static **Process Part** $\mathscr{P}$ and a dynamic **Data Part** $\mathscr{C}$
- **Process Part** describes how the data from the data part is **read and written**
- **Data part** contains **database instance** and
  the set of currently **active process instances**



Data-aware Business Processes (DABPs) model

- A **DABP** $\mathscr{B}$ consists of two parts:
  a static **Process Part** $\mathscr{P}$ and a dynamic **Data Part** $\mathscr{C}$
- **Process Part** describes how the data from the data part is **read and written**
- **Data part** contains **database instance** and
  the set of currently **active process instances**



New Information in the system is brought with **new process instances**

Process Part in DABPs

- **Process Part** $\mathscr{P} = \langle N, L \rangle$ is defined with **process net** N and **data rules** L

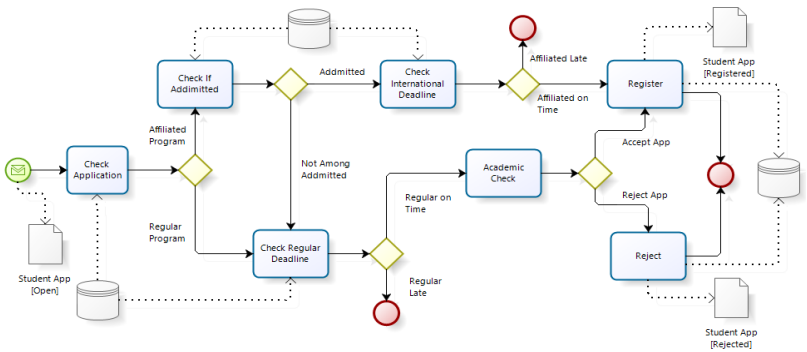
Process Part in DABPs

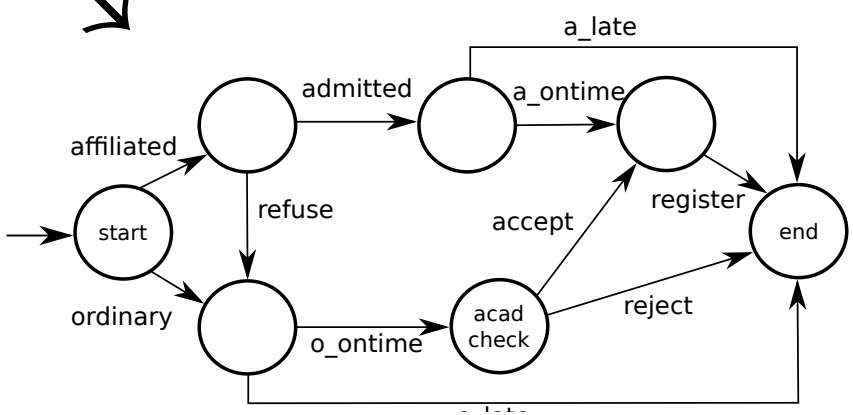
- **Process Part** $\mathscr{P} = \langle N, L \rangle$ is defined with **process net** N and **data rules** L
- **Process net** N is represented as a directed graph $\langle P, T \rangle$
  where T are transitions and P are places with a spacial place $start \in P$

Process Part in DABPs

- **Process Part** $\mathscr{P} = \langle N, L \rangle$ is defined with **process net** N and **data rules** L
- **Process net** N is represented as a directed graph $\langle P, T \rangle$
  where T are transitions and P are places with a spacial place $start \in P$

Example





Process Part in DABPs (2)

Data rules L labels every transition $t \in T$ with

Process Part in DABPs (2)

Data rules L labels every transition $t \in T$ with

- **Execution condition** E_t, a Boolean query that conditions the traversal of t, and

Process Part in DABPs (2)

Data rules L labels every transition $t \in T$ with

- **Execution condition** E_t, a Boolean query that conditions the traversal of t, and
- **Writing rule** $W_t = Q_t(\bar{x}) \rightarrow R(\bar{x})$, a rule that specifies which data is written

Process Part in DABPs (2)

Data rules L labels every transition $t \in T$ with

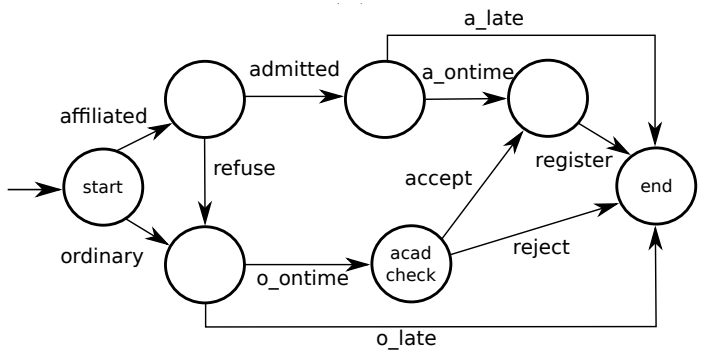
- **Execution condition** E_t, a Boolean query that conditions the traversal of t, and
- **Writing rule** $W_t = Q_t(\bar{x}) \to R(\bar{x})$, a rule that specifies which data is written
- Here, E_t and Q_t are **CQ with safe negation** over signature $\Sigma \cup I$ and $R \in \Sigma$

Process Part in DABPs (2)

Data rules L labels every transition $t \in T$ with

- **Execution condition** E_t, a Boolean query that conditions the traversal of t, and
- **Writing rule** $W_t = Q_t(\bar{x}) \to R(\bar{x})$, a rule that specifies which data is written
- Here, E_t and Q_t are **CQ with safe negation** over signature $\Sigma \cup I$ and $R \in \Sigma$
- Relation I is the input relation that describes process instance
  - e.g., student application form I('J. Smith','EMCL','Thursday 20$^{\text{th}}$ October, 2016')
    (the last I-argument is reserved for the instance timestamp)

Running example: process part



Transition	*Execution Condition* E_t
affiliated	$I(s,p,\tau),\ \mathit{StudyPlan}(p,\text{'affil'},m)$
ordinary	$I(s,p,\tau),\ \mathit{StudyPlan}(p,\text{'ord'},m),\ \neg\mathit{StudyPlan}(s,\text{'affil'},p)$
admitted	$I(s,p,\tau),\ \mathit{Admitted}(s,p)$
refuse	$I(s,p,\tau),\ \mathit{Admitted}(s,p),\ \mathit{StudyPlan}(p,\text{'ord'},m)$
a_late	$I(s,p,\tau),\ \mathit{Deadline}(\text{'affil'},d),\ \tau > d$
a_ontime	$I(s,p,\tau),\ \mathit{Deadline}(\text{'affil'},d),\ \tau < d$
o_late	$I(s,p,\tau),\ \mathit{Deadline}(\text{'affil'},d),\ \tau > d$
o_late	$I(s,p,\tau),\ \mathit{Deadline}(\text{'ord'},d),\ \tau > d$
o_ontime	$I(s,p,\tau),\ \mathit{Deadline}(\text{'ord'},d),\ \tau < d$
	Writing Rule W_t
register	$I(s,p,\tau),\ \mathit{StudyPlan}(p,r,m) \rightarrow \mathit{Registered}(s,m,p)$

Data Part in DABPs

Data Part $\mathscr{C}$, called **configuration**, is defined with $\langle D, O, M, \tau \rangle$ where

Data Part in DABPs

Data Part $\mathscr{C}$, called **configuration**, is defined with $\langle D, O, M, \tau \rangle$ where

- **Database instance** D over schema Σ

Data Part in DABPs

Data Part $\mathscr{C}$, called **configuration**, is defined with $\langle D, O, M, \tau \rangle$ where

- **Database instance** D over schema Σ
- Set of **process instances** O (called **data objects**)

Data Part in DABPs

Data Part $\mathscr{C}$, called **configuration**, is defined with $\langle D, O, M, \tau \rangle$ where

- **Database instance** D over schema Σ
- Set of **process instances** O (called **data objects**)
- **Mapping function** M that for every data object $o \in O$
  determines **current place** in the process $M_P(o) = p \in P$ and
  a single I-record $M_S(o) = I(\bar{s})$ of the input relation $I \notin \Sigma$
- **Current timestamp** τ of the configuration

Running Example: Initial Configuration

- Database instance D

StudyPlan

program	*registr.*	*master*
emSE	*affil*	mscCS
emCL	*affil*	mscCS
emCL	*reg*	mscCS
econ	*reg*	mscECO

Admitted

student	*program*
bob	emCL
mary	emSE

Deadline

registr.	*date*
reg	1^{st} Oct
affil	1^{st} Dec

Registered

student	*master*	*program*
bob	mscCS	emCL

Running Example: Initial Configuration

- Database instance D

StudyPlan

program	*registr.*	*master*
emSE	*affil*	mscCS
emCL	*affil*	mscCS
emCL	*reg*	mscCS
econ	*reg*	mscECO

Admitted

student	*program*
bob	emCL
mary	emSE

Deadline

registr.	*date*
reg	1^{st} Oct
affil	1^{st} Dec

Registered

student	*master*	*program*
bob	mscCS	emCL

- Data objects $O = \{o_1, o_2, o_3\}$

Running Example: Initial Configuration

- Database instance D

StudyPlan

program	*registr.*	*master*
emSE	*affil*	mscCS
emCL	*affil*	mscCS
emCL	*reg*	mscCS
econ	*reg*	mscECO

Admitted

student	*program*
bob	emCL
mary	emSE

Deadline

registr.	*date*
reg	1^{st} Oct
affil	1^{st} Dec

Registered

student	*master*	*program*
bob	mscCS	emCL

- Data objects $O = \{o_1, o_2, o_3\}$
- Mapping M

Mapping

id	*I*-record	place
o_3	(john, db, τ_3)	*start*
o_2	(alice, econ, τ_2)	*end*
o_1	(bob, emCL, τ_1)	*end*

Running Example: Initial Configuration

- Database instance D

StudyPlan

program	*registr.*	*master*
emSE	*affil*	mscCS
emCL	*affil*	mscCS
emCL	*reg*	mscCS
econ	*reg*	mscECO

Admitted

student	*program*
bob	emCL
mary	emSE

Deadline

registr.	*date*
reg	1^{st} Oct
affil	1^{st} Dec

Registered

student	*master*	*program*
bob	mscCS	emCL

- Data objects $O = \{o_1, o_2, o_3\}$
- Mapping M

Mapping

id	*I*-record	place
o_3	(john, db, τ_3)	*start*
o_2	(alice, econ, τ_2)	*end*
o_1	(bob, emCL, τ_1)	*end*

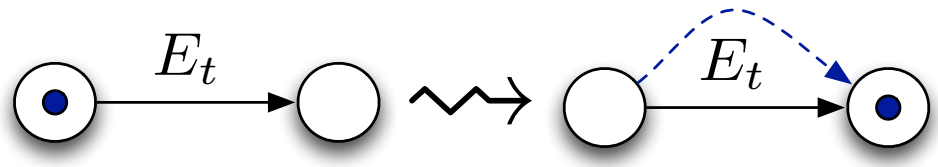
- Current time

$\tau =$ 'Thursday 20^{th} October, 2016'

Execution of DABPs

There are two kinds of **atomic execution** in DABPs

- **Traversal** of a transition in the net by an object



An object o with I-record $I(\bar{s})$ can traverse transition t if $E_t(D \cup I(\bar{s})) = \mathit{true}$
$+$ **the database instance** is updated so $D' = D \cup W_t(D \cup I(\bar{s}))$

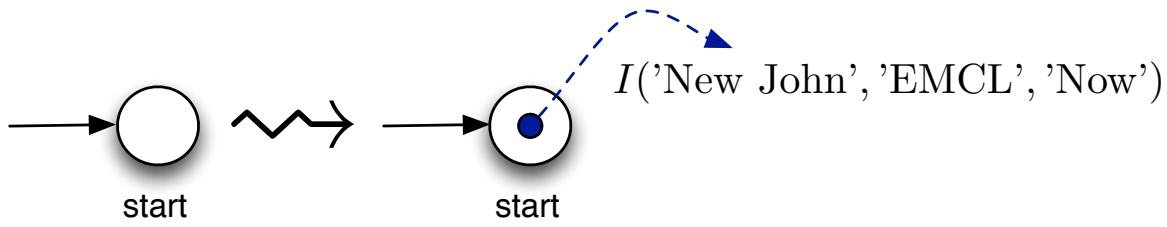
Execution of DABPs

There are two kinds of **atomic execution** in DABPs

- **Traversal** of a transition in the net by an object

  An object o with I-record $I(\bar{s})$ can traverse transition t if $E_t(D \cup I(\bar{s})) = true$
  + **the database instance** is updated so $D' = D \cup W_t(D \cup I(\bar{s}))$

- **Introduction** of a fresh object o with a fresh I-record

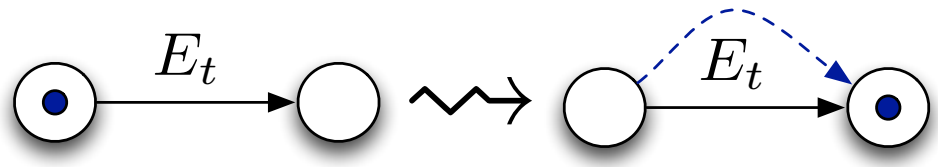


  + **the configuration timestamp** τ' is set to be the timestamps of o

Execution of DABPs

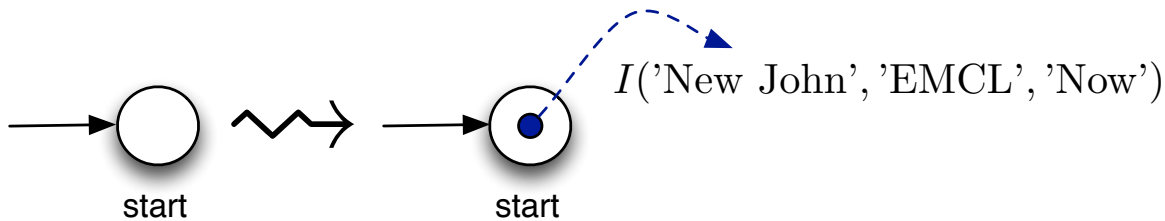
There are two kinds of **atomic execution** in DABPs

- **Traversal** of a transition in the net by an object



  An object o with I-record $I(\bar{s})$ can traverse transition t if $E_t(D \cup I(\bar{s})) = \mathit{true}$
  + **the database instance** is updated so $D' = D \cup W_t(D \cup I(\bar{s}))$

- **Introduction** of a fresh object o with a fresh I-record



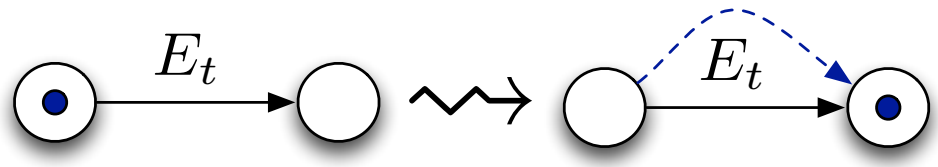
  + **the configuration timestamp** τ' is set to be the timestamps of o

- In both cases a **new configuration** $\mathscr{C}' = \langle D', O', M', \tau' \rangle$ is obtained as a result

Execution of DABPs

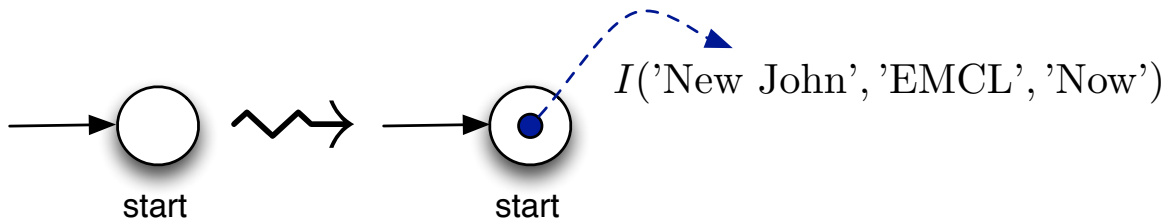
There are two kinds of **atomic execution** in DABPs

- **Traversal** of a transition in the net by an object



  An object o with I-record $I(\bar{s})$ can traverse transition t if $E_t(D \cup I(\bar{s})) = \mathit{true}$
  + **the database instance** is updated so $D' = D \cup W_t(D \cup I(\bar{s}))$

- **Introduction** of a fresh object o with a fresh I-record



  + **the configuration timestamp** τ' is set to be the timestamps of o

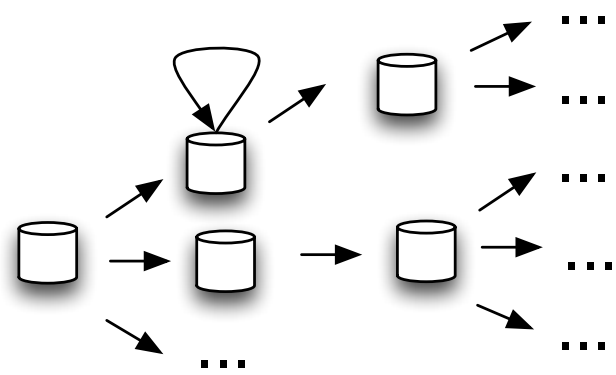
- In both cases a **new configuration** $\mathscr{C}' = \langle D', O', M', \tau' \rangle$ is obtained as a result
- **Executions** in DABPs are finite sequences of atomic executions

Table of Contents

Query Stability Formally

- Query Q is **stable** in DABP $\mathscr{B} = \langle \mathscr{P}, \mathscr{C} \rangle$ with database D if for any *reachable* configuration with the database D'

$$Q(D) = Q(D')$$



Running Example: Stability of Queries

- Q_{cs} : Who are the registered students at the Faculty of Computer Science?
  e.g., $Q_{cs}(x) \leftarrow Registered(x, \text{'mscCS'}, p)$
- Q_{eco} : Who are the registered students at the Faculty of Economics?
  e.g., $Q_{eco}(x) \leftarrow Registered(x, \text{'mscECO'}, p)$

Faculty of Economics	Student places 2012 / 2013	Enrollments 2011 / 2012	Enrollments 2012 / 2013	%	Stable?	< 1st Oct	1st Oct-1st Dec	> 1st Dec
Bachelor in Economics and Management	[redacted]	[redacted]	[redacted]	20,78%		NO!	YES!	YES!
Bachelor in Tourism, Sport and Event Management	[redacted]	[redacted]	[redacted]	74,51%		NO!	YES!	YES!
Bachelor in Economics and Social Sciences	[redacted]	[redacted]	[redacted]	17,24%		NO!	YES!	YES!
Master in Entrepreneurship and Innovation	[redacted]	[redacted]	[redacted]	114,29%		NO!	YES!	YES!
Master in Economics and Management of the public sector	[redacted]	[redacted]	[redacted]	22,73%		NO!	YES!	YES!
Sum	[redacted]	[redacted]	[redacted]	41,45%		NO!	YES!	YES!

Faculty of Computer Science	Student Places 2012 / 2013	Enrollments 2011 / 2012	Enrollments 2012 / 2013	%		< 1st Oct	1st Oct-1st Dec	> 1st Dec
Bachelor in Computer Science and Engineering	[redacted]	[redacted]	[redacted]	30,43%		NO!	YES!	YES!
Master of Science in Computer Science	[redacted]	[redacted]	[redacted]	-44,44%		NO!	NO!	YES!
PhD in Computer Science	[redacted]	[redacted]	[redacted]	#VALUE!		NO!	NO!	YES!
Sum	[redacted]	[redacted]	[redacted]	-25,37%		NO!	NO!	YES!

Table of Contents

Reasoning about Query Stability in DABP

- We studied several types of processes types that differ in

Reasoning about Query Stability in DABP

- We studied several types of processes types that differ in
- **Semantics: Open or Closed**

Reasoning about Query Stability in DABP

- We studied several types of processes types that differ in
- **Semantics: Open or Closed**
  - **Open semantics**: new process instance can start at any moment

Reasoning about Query Stability in DABP

- We studied several types of processes types that differ in
- **Semantics: Open or Closed**
  - **Open semantics**: new process instance can start at any moment
  - **Closed semantics**: no new process instance can be started,
    thus only "unfinished" objects can impact stability

Reasoning about Query Stability in DABP

- We studied several types of processes types that differ in
- **Semantics: Open or Closed**
  - **Open semantics**: new process instance can start at any moment
  - **Closed semantics**: no new process instance can be started,
    thus only "unfinished" objects can impact stability
- **Initial Configuration: Fresh or Arbitrary**

Reasoning about Query Stability in DABP

- We studied several types of processes types that differ in
- **Semantics: Open or Closed**
  - **Open semantics**: new process instance can start at any moment
  - **Closed semantics**: no new process instance can be started,
    thus only "unfinished" objects can impact stability
- **Initial Configuration: Fresh or Arbitrary**
  - **Fresh**: the configuration does not contain any object

Reasoning about Query Stability in DABP

- We studied several types of processes types that differ in
- **Semantics: Open or Closed**
  - **Open semantics**: new process instance can start at any moment
  - **Closed semantics**: no new process instance can be started,
    thus only "unfinished" objects can impact stability
- **Initial Configuration: Fresh or Arbitrary**
  - **Fresh**: the configuration does not contain any object
  - **Arbitrary**: no assumption on the presence or absence of objects

Reasoning about Query Stability in DABP

- We studied several types of processes types that differ in
- **Semantics: Open or Closed**
  - **Open semantics**: new process instance can start at any moment
  - **Closed semantics**: no new process instance can be started, thus only "unfinished" objects can impact stability
- **Initial Configuration: Fresh or Arbitrary**
  - **Fresh**: the configuration does not contain any object
  - **Arbitrary**: no assumption on the presence or absence of objects
- **Process Net: Cyclic or Acylcic**

Reasoning about Query Stability in DABP

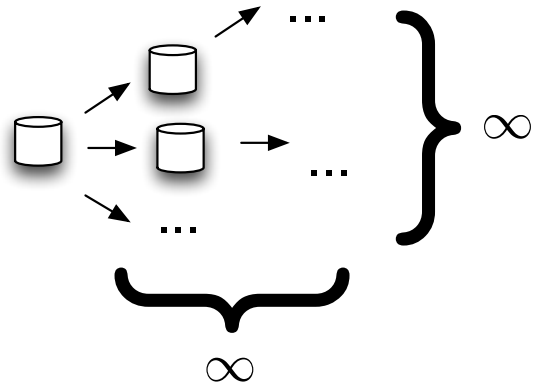
- We studied several types of processes types that differ in
- **Semantics: Open or Closed**
  - **Open semantics**: new process instance can start at any moment
  - **Closed semantics**: no new process instance can be started, thus only "unfinished" objects can impact stability
- **Initial Configuration: Fresh or Arbitrary**
  - **Fresh**: the configuration does not contain any object
  - **Arbitrary**: no assumption on the presence or absence of objects
- **Process Net: Cyclic or Acylcic**
- **Process Rules: Normal**(w/ negation) or **or Positive**(w/o negation)

How complex is to check stability for Conjunctive Queries

- **Undcidable** in the most general case :(
  even in the **data** and **query** complexity

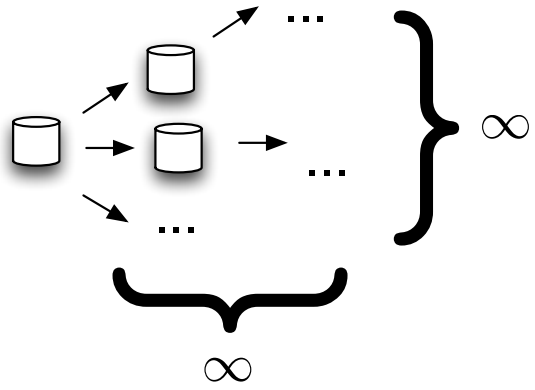
How complex is to check stability for Conjunctive Queries

- **Undcidable** in the most general case :(
  even in the **data** and **query** complexity



How complex is to check stability for Conjunctive Queries

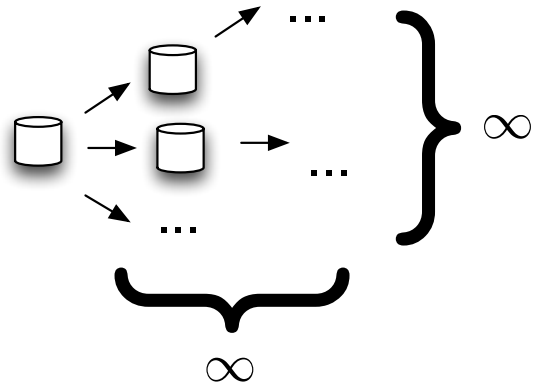
- **Undcidable** in the most general case :(
  even in the **data** and **query** complexity



- Still, many **decidable** cases :) by restricting DAPBs to

How complex is to check stability for Conjunctive Queries

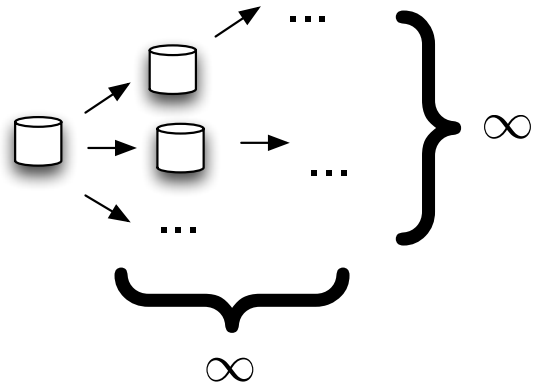
- **Undcidable** in the most general case :(
  even in the **data** and **query** complexity



- Still, many **decidable** cases :) by restricting DAPBs to
  - **closed semantics** (no fresh instances are allowed, so obvious :)

How complex is to check stability for Conjunctive Queries

- **Undcidable** in the most general case :(
  even in the **data** and **query** complexity



- Still, many **decidable** cases :) by restricting DAPBs to
  - **closed semantics** (no fresh instances are allowed, so obvious :)
  - **positive rules** (little less obvious)

How complex is to check stability for Conjunctive Queries

Process Rules	Process Net	Semantics	Configuration	Data	Process	Query	Combined
Normal	*Cyclic / Acyclic*	*Open*	*Arbitrary*	UNDEC.	UNDEC.	UNDEC.	UNDEC.
	Cyclic / Acyclic	*Open*	*Fresh*	UNDEC.	UNDEC.	UNDEC.	UNDEC.
	Cyclic	*Closed*	*Arbitrary*	CO-NP	CO-NEXPTIME	Π_2^P	CO-NEXPTIME
	Acyclic	*Closed*	*Arbitrary*	CO-NP	PSPACE	Π_2^P	PSPACE
Positive	*Cyclic / Acyclic*	*Open*	*Arbitrary*	CO-NP	EXPTIME	Π_2^P	EXPTIME
	Cyclic / Acyclic	*Open*	*Fresh*	PTIME	EXPTIME	Π_2^P	EXPTIME
	Cyclic	*Closed*	*Arbitrary*	CO-NP	EXPTIME	Π_2^P	EXPTIME
	Acyclic	*Closed*	*Arbitrary*	CO-NP	PSPACE	Π_2^P	PSPACE

How complex is to check stability for Conjunctive Queries

Process Rules	Process Net	Semantics	Configuration	Data	Process	Query	Combined
Normal	*Cyclic / Acyclic*	*Open*	*Arbitrary*	UNDEC.	UNDEC.	UNDEC.	UNDEC.
	Cyclic / Acyclic	*Open*	*Fresh*	UNDEC.	UNDEC.	UNDEC.	UNDEC.
	Cyclic	*Closed*	*Arbitrary*	co-NP	co-NEXPTIME	Π_2^P	co-NEXPTIME
	Acyclic	*Closed*	*Arbitrary*	co-NP	PSPACE	Π_2^P	PSPACE
Positive	*Cyclic / Acyclic*	*Open*	*Arbitrary*	co-NP	EXPTIME	Π_2^P	EXPTIME
	Cyclic / Acyclic	*Open*	*Fresh*	PTIME	EXPTIME	Π_2^P	EXPTIME
	Cyclic	*Closed*	*Arbitrary*	co-NP	EXPTIME	Π_2^P	EXPTIME
	Acyclic	*Closed*	*Arbitrary*	co-NP	PSPACE	Π_2^P	PSPACE

Encoding techniques for decidable DABPs

- Even for **positive** or **closed semantics** we may have unbounded executions

Encoding techniques for decidable DABPs

- Even for **positive** or **closed semantics** we may have unbounded executions
- **Abstraction principle** for **positive DABPs** under **open semantics**:
  when checking stability it is enough to consider at most **one fresh constant**

Encoding techniques for decidable DABPs

- Even for **positive** or **closed semantics** we may have unbounded executions
- **Abstraction principle** for **positive DABPs** under **open semantics**:
  when checking stability it is enough to consider at most **one fresh constant**
- Now, only **cycles** may produce unbounded executions

Encoding techniques for decidable DABPs

- Even for **positive** or **closed semantics** we may have unbounded executions
- **Abstraction principle** for **positive DABPs** under **open semantics**:
  when checking stability it is enough to consider at most **one fresh constant**
- Now, only **cycles** may produce unbounded executions
- But we can produce at most **exponentially many facts**,
  in particular at most $const(\mathscr{B})^{sig(\mathscr{B})}$ many

Encoding techniques for decidable DABPs

- Even for **positive** or **closed semantics** we may have unbounded executions
- **Abstraction principle** for **positive DABPs** under **open semantics**:
  when checking stability it is enough to consider at most **one fresh constant**
- Now, only **cycles** may produce unbounded executions
- But we can produce at most **exponentially many facts**,
  in particular at most $const(\mathscr{B})^{sig(\mathscr{B})}$ many
- Thus we need consider only **exponentially long executions**

Encoding techniques for decidable DABPs

- Even for **positive** or **closed semantics** we may have unbounded executions
- **Abstraction principle** for **positive DABPs** under **open semantics**:
  when checking stability it is enough to consider at most **one fresh constant**
- Now, only **cycles** may produce unbounded executions
- But we can produce at most **exponentially many facts**,
  in particular at most $\mathit{const}(\mathscr{B})^{\mathit{sig}(\mathscr{B})}$ many
- Thus we need consider only **exponentially long executions**

We can guess an exponential long execution using **Datalog** under stable model semantics (co-NExpTime)

Encoding techniques for decidable DABPs

- Even for **positive** or **closed semantics** we may have unbounded executions
- **Abstraction principle** for **positive DABPs** under **open semantics**:
  when checking stability it is enough to consider at most **one fresh constant**
- Now, only **cycles** may produce unbounded executions
- But we can produce at most **exponentially many facts**,
  in particular at most $\mathit{const}(\mathscr{B})^{\mathit{sig}(\mathscr{B})}$ many
- Thus we need consider only **exponentially long executions**

We can guess an exponential long execution using **Datalog** under stable model semantics (co-NExpTime)

Encoding techniques for decidable DABPs

- Even for **positive** or **closed semantics** we may have unbounded executions
- **Abstraction principle** for **positive DABPs** under **open semantics**:
  when checking stability it is enough to consider at most **one fresh constant**
- Now, only **cycles** may produce unbounded executions
- But we can produce at most **exponentially many facts**,
  in particular at most $const(\mathscr{B})^{sig(\mathscr{B})}$ many
- Thus we need consider only **exponentially long executions**

We can guess an exponential long execution using **Datalog** under stable model semantics (co-NExpTime)

Theorem
Let $\omega = o_1, t_1, o_2, t_2, \ldots, o_n, t_n$ be an execution in $\mathscr{B}$. Then one can construct a Datalog program $\Pi_{\mathscr{B}}$ such that

Encoding techniques for decidable DABPs

- Even for **positive** or **closed semantics** we may have unbounded executions
- **Abstraction principle** for **positive DABPs** under **open semantics**:
  when checking stability it is enough to consider at most **one fresh constant**
- Now, only **cycles** may produce unbounded executions
- But we can produce at most **exponentially many facts**,
  in particular at most $\mathit{const}(\mathscr{B})^{\mathit{sig}(\mathscr{B})}$ many
- Thus we need consider only **exponentially long executions**

We can guess an exponential long execution using **Datalog** under stable model semantics (co-NExpTime)

Theorem
Let $\omega = o_1, t_1, o_2, t_2, \ldots, o_n, t_n$ *be an execution in* $\mathscr{B}$. *Then one can construct a Datalog program* $\Pi_{\mathscr{B}}$ *such that*

- *the execution* ω *generates ground atoms* $R_1(\bar{s}_1), \ldots, R_n(\bar{s}_n)$, ***iff***

Encoding techniques for decidable DABPs

- Even for **positive** or **closed semantics** we may have unbounded executions
- **Abstraction principle** for **positive DABPs** under **open semantics**:
  when checking stability it is enough to consider at most **one fresh constant**
- Now, only **cycles** may produce unbounded executions
- But we can produce at most **exponentially many facts**,
  in particular at most $\mathit{const}(\mathscr{B})^{\mathit{sig}(\mathscr{B})}$ many
- Thus we need consider only **exponentially long executions**

We can guess an exponential long execution using **Datalog** under stable model semantics (co-NExpTime)

Theorem
Let $\omega = o_1, t_1, o_2, t_2, \ldots, o_n, t_n$ be an execution in $\mathscr{B}$. Then one can construct a Datalog program $\Pi_{\mathscr{B}}$ such that

- *the execution ω generates ground atoms $R_1(\bar{s}_1), \ldots, R_n(\bar{s}_n)$,* ***iff***
- $\Pi_{\mathscr{B}} \models_{\mathit{brave}} \tilde{R}_1(\bar{\omega}, \bar{s}_1), \ldots, \tilde{R}_n(\bar{\omega}, \bar{s}_n)$

Encoding techniques for decidable DABPs (2)

- **Acyclic DABPs under open semantics**:
  it is sufficient to consider polynomially many exponential executions

Encoding techniques for decidable DABPs (2)

- **Acyclic DABPs under open semantics**:
  it is sufficient to consider polynomially many exponential executions

Compute all executions using **recursive** positive **Datalog** (ExpTime)

Encoding techniques for decidable DABPs (2)

- **Acyclic DABPs under open semantics**:
  it is sufficient to consider polynomially many exponential executions

Compute all executions using **recursive** positive **Datalog** (ExpTime)

- **Acyclic DABPs under closed Semantics**: only polynomially long executions

Encoding techniques for decidable DABPs (2)

- **Acyclic DABPs under open semantics**:
  it is sufficient to consider polynomially many exponential executions

  ➥ Compute all executions using **recursive** positive **Datalog** (ExpTime)

- **Acyclic DABPs under closed Semantics**: only polynomially long executions

  ➥ Compute all executions using **nonrecursive Datalog** (PSpace)

Encoding techniques for decidable DABPs (2)

- **Acyclic DABPs under open semantics**:
  it is sufficient to consider polynomially many exponential executions


Compute all executions using **recursive** positive **Datalog** (ExpTime)

- **Acyclic DABPs under closed Semantics**: only polynomially long executions


Compute all executions using **nonrecursive Datalog** (PSpace)


Can we obtain interesting **tractable** cases?

Read-Only Write-only DAPBs

- Split schema Σ into **Read-only** schema Σ_R and **Write-Only** schema Σ_W

Read-Only Write-only DAPBs

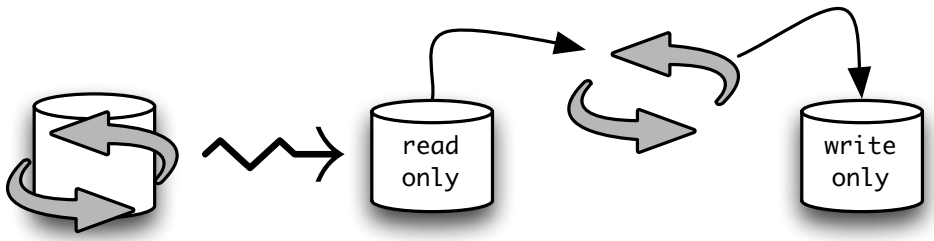
- Split schema Σ into **Read-only** schema Σ_R and **Write-Only** schema Σ_W
- *Execution conditions* and *bodies of writing rules* are over Σ_R

Read-Only Write-only DAPBs

- Split schema Σ into **Read-only** schema Σ_R and **Write-Only** schema Σ_W
- *Execution conditions* and *bodies of writing rules* are over Σ_R
- *Heads of writing rules* are over Σ_W

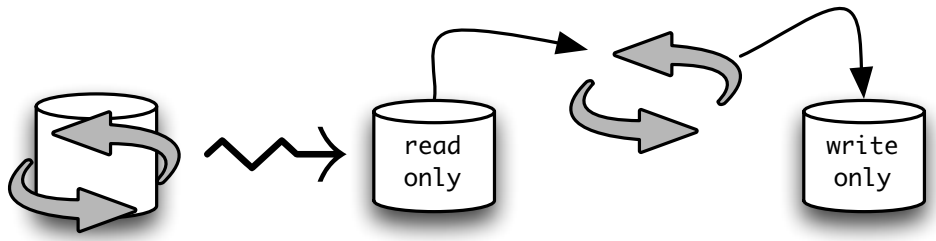
Read-Only Write-only DAPBs

- Split schema Σ into **Read-only** schema Σ_R and **Write-Only** schema Σ_W
- *Execution conditions* and *bodies of writing rules* are over Σ_R
- *Heads of writing rules* are over Σ_W



Read-Only Write-only DAPBs

- Split schema Σ into **Read-only** schema Σ_R and **Write-Only** schema Σ_W
- *Execution conditions* and *bodies of writing rules* are over Σ_R
- *Heads of writing rules* are over Σ_W



- *Thus, objects cannot read what they have written*

Checking stability in Read-Only Write-only DAPBs

- nicer complexities :)

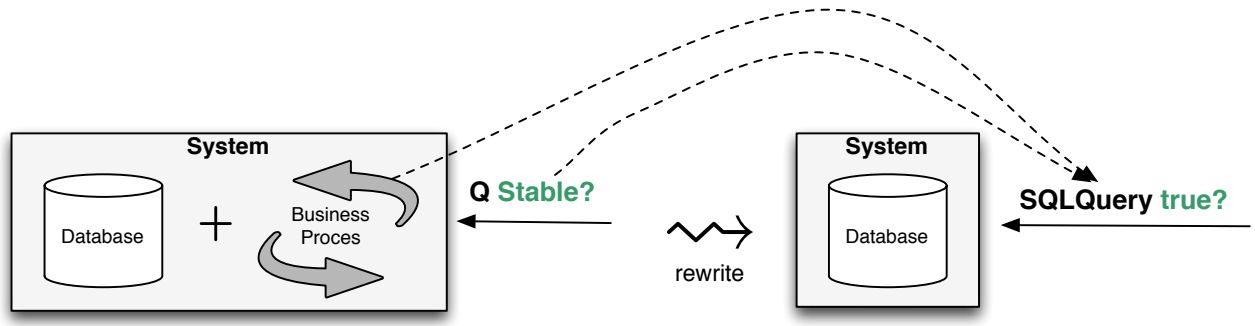
Process Rules	Process Net	Semantics	Configuration	Data	Process	Query	Combined
Normal	*Cyclic / Acyclic*	*Open*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P
	Cyclic / Acyclic	*Open*	*Fresh*	in AC^0	co-NP	Π_2^P	Π_2^P
	Cyclic	*Closed*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P
	Acyclic	*Closed*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P
Positive	*Cyclic / Acyclic*	*Open*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P
	Cyclic / Acyclic	*Open*	*Fresh*	in AC^0	co-NP	Π_2^P	Π_2^P
	Cyclic	*Closed*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P
	Acyclic	*Closed*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P

Checking stability in Read-Only Write-only DAPBs

- **nicer complexities :)**

Process Rules	Process Net	Semantics	Configuration	Data	Process	Query	Combined
Normal	*Cyclic / Acyclic*	*Open*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P
	Cyclic / Acyclic	*Open*	*Fresh*	in AC^0	co-NP	Π_2^P	Π_2^P
	Cyclic	*Closed*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P
	Acyclic	*Closed*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P
Positive	*Cyclic / Acyclic*	*Open*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P
	Cyclic / Acyclic	*Open*	*Fresh*	in AC^0	co-NP	Π_2^P	Π_2^P
	Cyclic	*Closed*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P
	Acyclic	*Closed*	*Arbitrary*	in AC^0	co-NP	Π_2^P	Π_2^P

- Checking Stability in rowo DABPs is **FO-rewritable**



Future Work

We plan to consider

- *More expressive queries*, e.g., CQ with negation or FO;
- *Stability of aggregate queries* / *aggregates in the process rules*;
- *Quantify instability*,
  e.g., compute the minimal/maximal number of new answers;
- Other data quality aspects such as *data timeliness* and *data currency*.

Thank you!