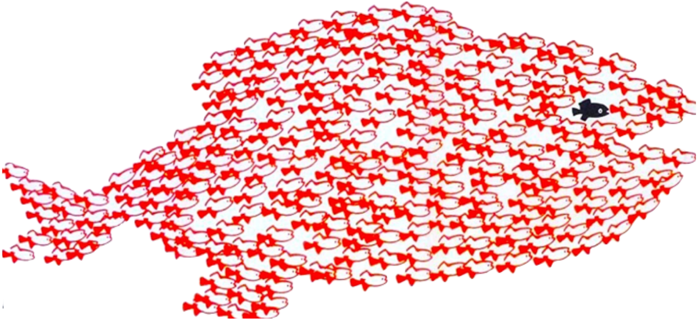
Putting Decisions
in Perspective(s)

Marco Montali

Free University of Bozen-Bolzano

DEC2H 2019, Vienna, Austria



Decision Models Strike Back

Decision Model and Notation (**DMN**) standard by OMG:

- Elicitation and clean representation of decision models.
- Decision: set of business rules for a single decision with fixed input/output attributes. Shown in a **table**.
- Decision Requirements Graph: network of decisions, binding their input/outputs to obtain a more complex decision. Shown in a **decision requirement diagram**.

Wide adoption by the industry.

Success Factor #1: Timeliness

Organizations are increasingly process-oriented.

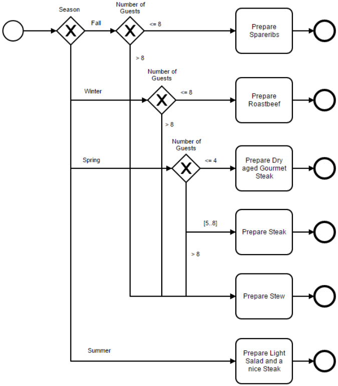
- DMN encourages separation of concerns between the process logic and the decision logic.
- Clarity, modularity, reusability.

Success Factor #1: Timeliness

Organizations are increasingly process-oriented.

- DMN encourages separation of concerns between the process logic and the decision logic.
- Clarity, modularity, reusability.

From BPMN. . .

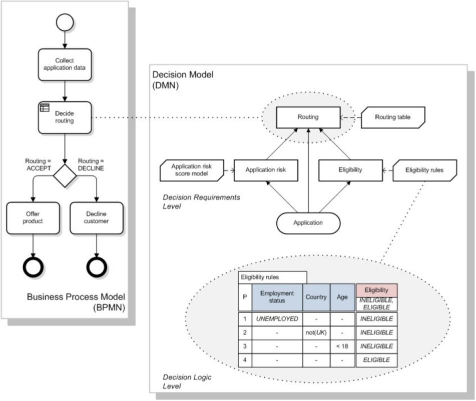


Success Factor #1: Timeliness

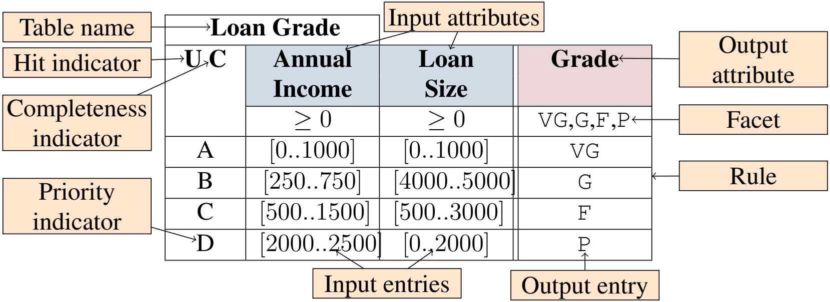
Organizations are increasingly process-oriented.

- DMN encourages separation of concerns between the process logic and the decision logic.
- Clarity, modularity, reusability.

. . . to BPMN+DMN



Success Factor #2: Understandability



Rule conditions specified using the Friendly Enough Expression Language, coming in two flavours:

- **S-FEEL** - simple and graphical.
- **FEEL** - powerful and textual.

Success Factor #2: Understandability

Table name · Hit indicator · Completeness indicator · Priority indicator · Input attributes · Output attribute · Facet · Rule · Input entries · Output entry

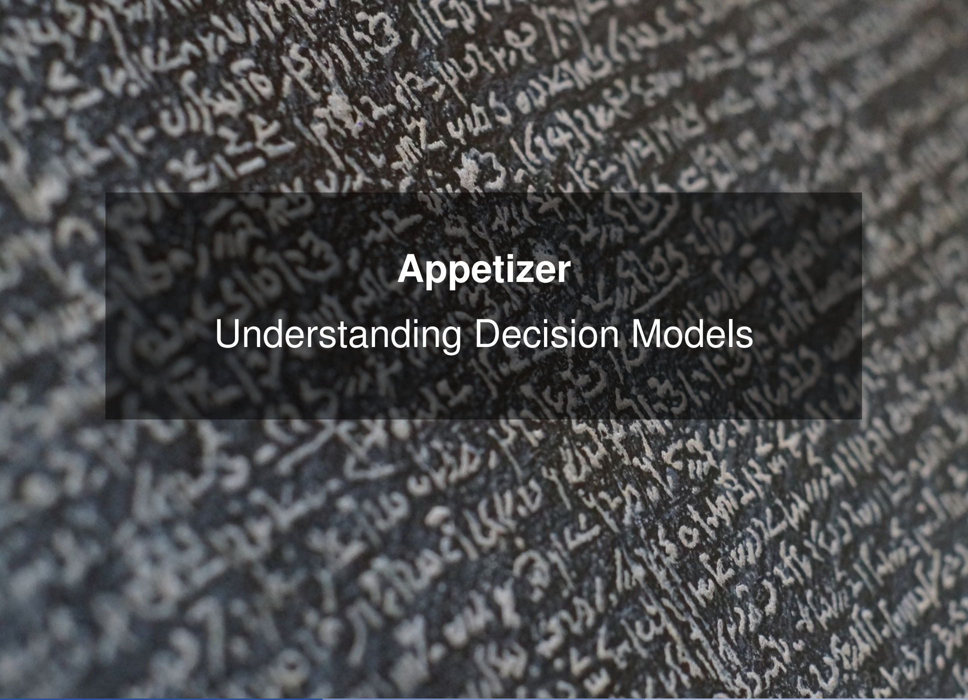
Loan Grade			
U,C	**Annual Income**	**Loan Size**	**Grade**
	≥ 0	≥ 0	VG,G,F,P
A	[0..1000]	[0..1000]	VG
B	[250..750]	[4000..5000]	G
C	[500..1500]	[500..3000]	F
D	[2000..2500]	[0..2000]	P

Rule conditions specified using the Friendly Enough Expression Language, coming in two flavours:

- **S-FEEL** - simple and graphical.
- **FEEL** - powerful and textual.

We focus on S-FEEL (with extensions)

The *controversial pearl* of the standard.



Appetizer

Understanding Decision Models

A Simple Decision Table

TurnAround is a courier company delivering packages with different transportation modalities, depending on the package physical features.

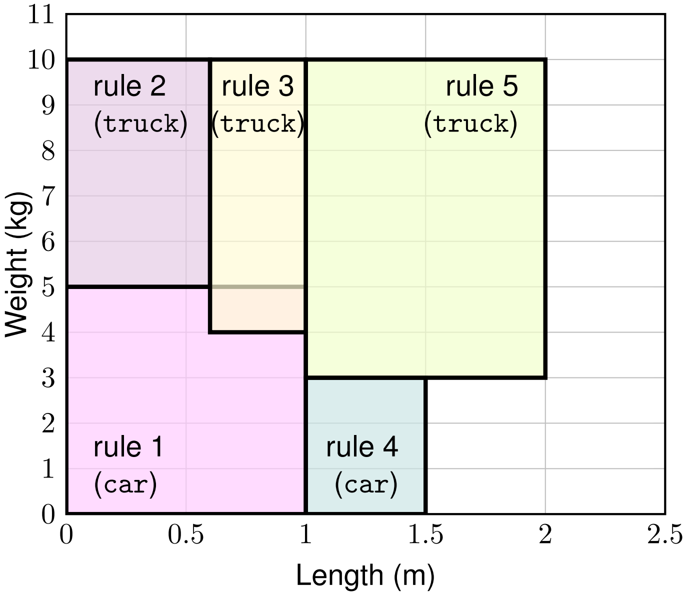
Decision logic for the shipment modality

Package Shipment			
P	Length (m)	Weight (kg)	ShipBy
	> 0	> 0	`car, truck`
1	(0.0,1.0]	(0, 5]	`car`
2	(0.0,0.6]	(5,10]	`truck`
3	(0.6,1.0]	(4,10]	`truck`
4	(1.0,1.5]	(0, 3]	`car`
5	(1.0,2.0]	(3,10]	`truck`

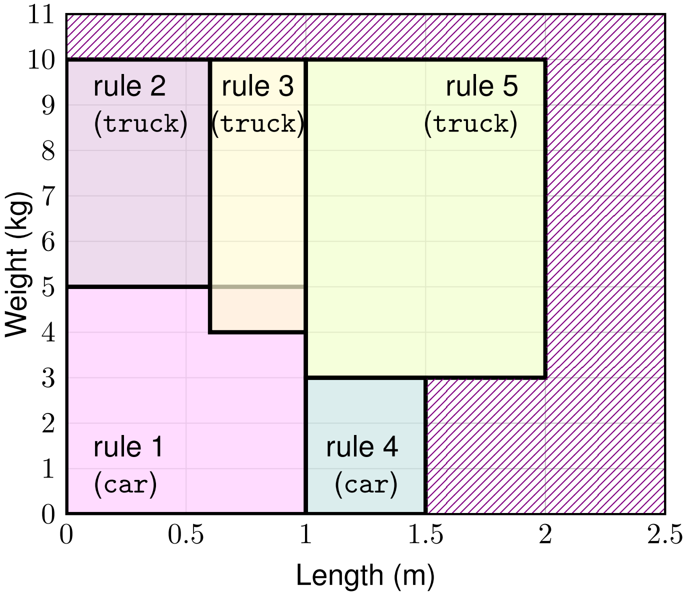
Question

What can we say about the *shipment modality* decision table?

Geometric Intuition



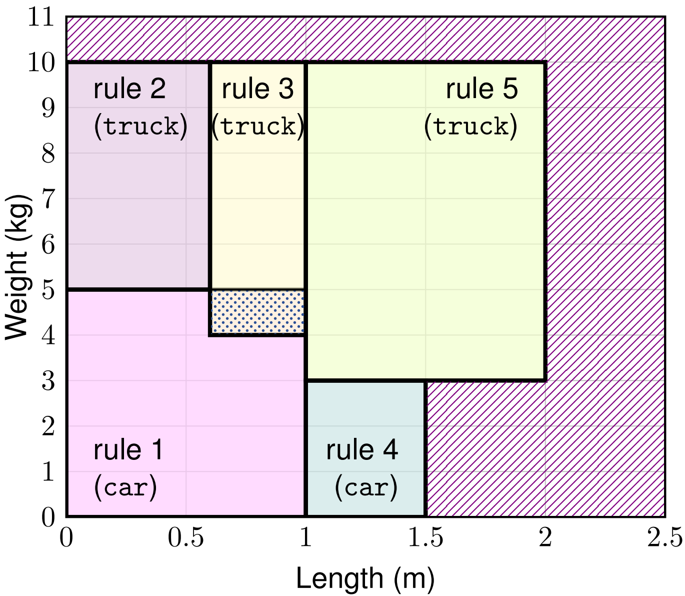
Geometric Intuition



Incomplete

Inputs with no matching rule.

Geometric Intuition



Incomplete

Inputs with no matching rule.

Overlaps

Inputs with multiple matching rules.

P: a reasonable hit policy.

Formalization and Reasoning [___,BPM2016;___,IS2018]

1. Logic-based semantics of S-FEEL DMN

2. Logic-based formalization of analysis tasks

3. Implementation

Formalization and Reasoning [___,BPM2016;___,IS2018]

1. Logic-based semantics of S-FEEL DMN

- Requires a prior uniqueification [Batoulis and Weske, BPMDemo2018] of the DMN table
- Rules become quantifier-free multi-sorted FOL formulae with datatypes and their comparison predicates.
- Tuple-based: rules induce an input/output relation over tuples of input/output values.

2. Logic-based formalization of analysis tasks

3. Implementation

Formalization and Reasoning [___,BPM2016;___,IS2018]

1. Logic-based semantics of S-FEEL DMN

2. Logic-based formalization of analysis tasks

Quantified formulae capturing table properties:

- compatibility between conditions and attribute facets;
- completeness;
- adequacy of hit policies (does the chosen hit policy reflect the table semantics?).

3. Implementation

Formalization and Reasoning [___,BPM2016;___,IS2018]

1. Logic-based semantics of S-FEEL DMN

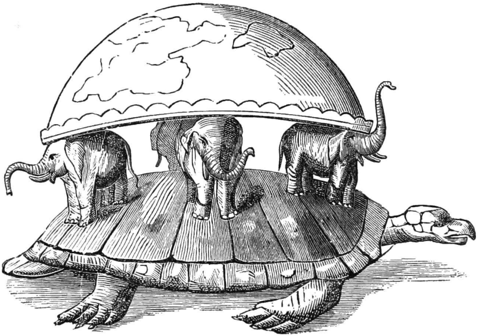
2. Logic-based formalization of analysis tasks

3. Implementation

- In principle, 1.+2. directly enable the usage of SMT solvers to analyze decision tables.
- In practice:
  - We interpret rules geometrically (hyperrectangles).
  - We take state-of-the art algorithms and use them for analysis and simplification of tables.
  - Impressive performance.

Decisions are not alone!

Organization



Strategic Management	Business Process Management	Master Data Management	Enterprise Decision Management
Goals and resources	Operational processes	Relevant facts	Decision logic

Putting decisions in perspective

Key questions

- How to integrate decision models within an organization?
- How does this impact the decision logic?
- Which analysis tasks emerge? Can they be solved?

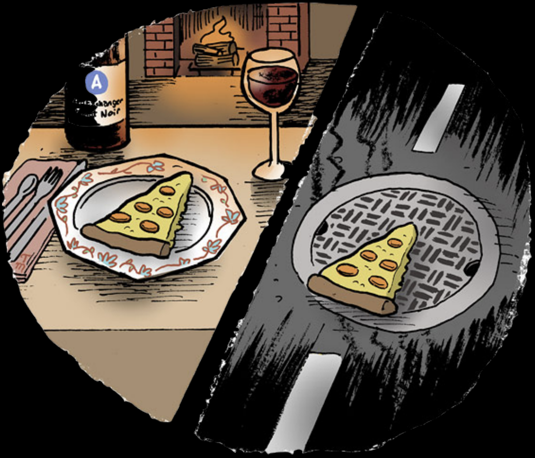
Putting decisions in perspective(s)

Key questions

- How to integrate decision models within an organization?
- How does this impact the decision logic?
- Which analysis tasks emerge? Can they be solved?

Two settings

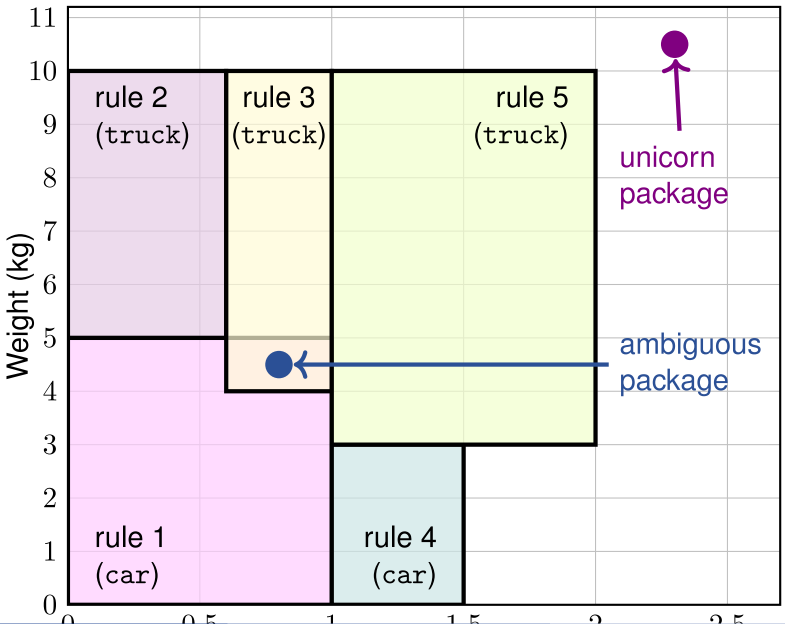
1. Decision tables in the context of background structural knowledge.
2. Processes routing cases based on decision tables.



First Course

Decision Models and Background Knowledge

Which Packages Exist?



Packages within an Organization

BLACKSHIP is a mediator company:

- ← Offers to customers package configurations. Helps customers in shipping their packages.
- → Interacts with a courier company for the actual delivery.

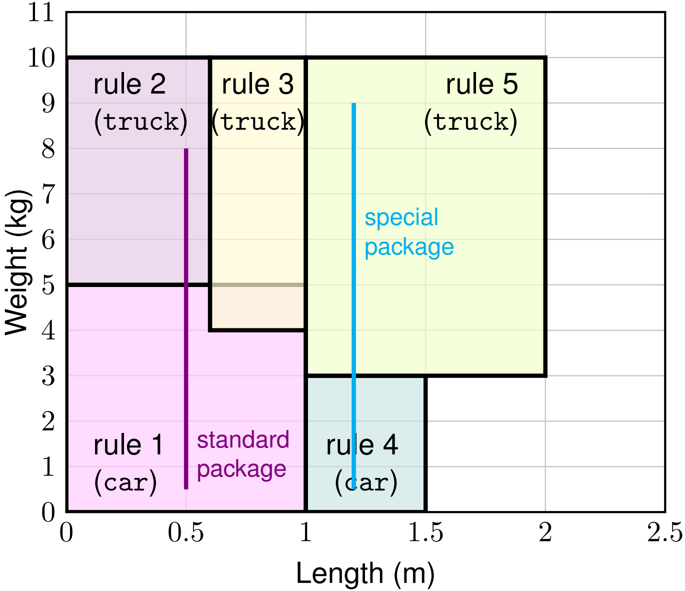
KB of packages offered by BLACKSHIP

A1	There are two types of packages: standard and special.
A2	Each package is either standard or special.
A3	The minimum weight for a package is 0.5 kg.
A4	A standard package has a length of 0.5 m and bears at most 8 kg.
A5	A special package has a length of 1.2 m and bears at most 9 kg.

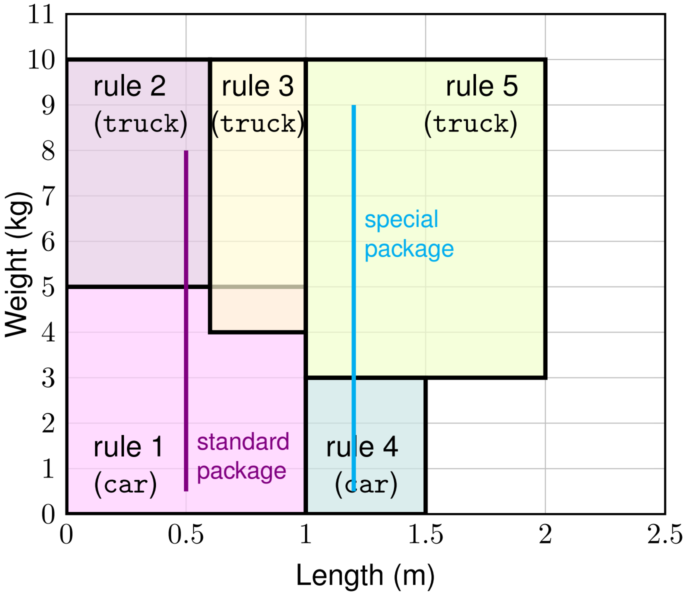
Question

What happens if BLACKSHIP selects TURNAROUND as partner?

Decision in the Context of Background Knowledge



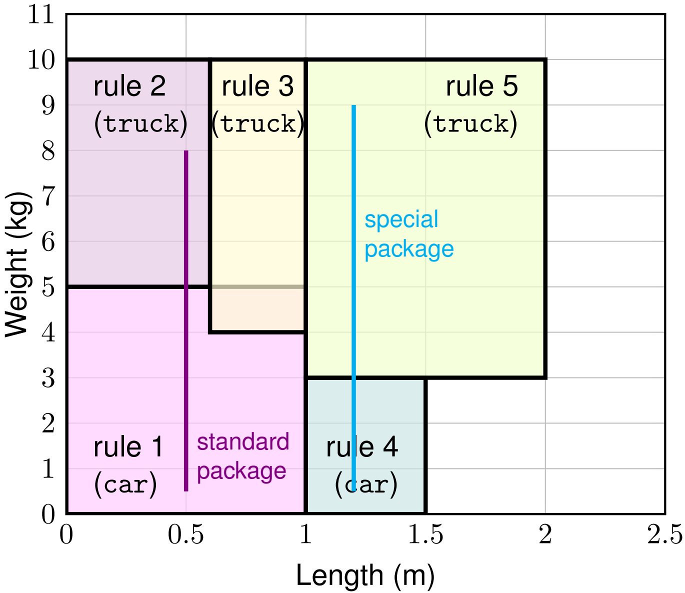
Decision in the Context of Background Knowledge



Complete

Standard and special packages always match with a rule.

Decision in the Context of Background Knowledge



Complete

Standard and special packages always match with a rule.

Unique hit

Standard and special packages always match with a single rule.

P: priority never applied.

A More Complex Example

Inspired by the Ship and Port Facility Security Code:

- Ship clearance in the Netherlands.
- March 2016 challenge at `dmcommunity.org`.

Knowledge of Ships

There are several types of ships, characterized by:

- length (in *m*);
- draft size (in *m*);
- capacity (in *TEU*).

Knowledge of Ships

There are several types of ships, characterized by:

- length (in *m*);
- draft size (in *m*);
- capacity (in *TEU*).

Ship KB

Ship Type	**Short**	**Length** (m)	**Draft** (m)	**Capacity** (TEU)
Converted Cargo Vessel	*CCV*	135	0 – 9	500
Converted Tanker	*CT*	200	0 – 9	800
Cellular Containership	*CC*	215	10	1000 – 2500
Small Panamax Class	*SPC*	250	11 – 12	3000
Large Panamax Class	*LPC*	290	11 – 12	4000
Post Panamax	*PP*	275 – 305	11 – 13	4000 – 5000
Post Panamax Plus	*PPP*	335	13 – 14	5000 – 8000
New Panamax	*NP*	397	15.5	11 000 – 14 500

Knowledge of Ships

There are several types of ships, characterized by:

- length (in *m*);
- draft size (in *m*);
- capacity (in *TEU*).

Ship KB

Ship Type	**Short**	**Length** (m)	**Draft** (m)	**Capacity** (TEU)
Converted Cargo Vessel	*CCV*	135	0 – 9	500
Converted Tanker	*CT*	200	0 – 9	800
Cellular Containership	*CC*	215	10	1000 – 2500
Small Panamax Class	*SPC*	250	11 – 12	3000
Large Panamax Class	*LPC*	290	11 – 12	4000
Post Panamax	*PP*	275 – 305	11 – 13	4000 – 5000
Post Panamax Plus	*PPP*	335	13 – 14	5000 – 8000
New Panamax	*NP*	397	15.5	11 000 – 14 500

Warning!

This is **not** a decision table. This is a set of **constraints** relating the ship types with corresponding possible dimensions.

Clearance Rules

A vessel may enter a port if:

- it is equipped with a valid certificate of registry;
- it meets the safety requirements.

Clearance Rules

A vessel may enter a port if:

- it is equipped with a valid certificate of registry;
- it meets the safety requirements.

Valid certificate of registry

Certificate expiration date $>$ current date.

Safety Requirements

Based on ship characteristics and the amount of residual cargo:

- small ships (with length ≤ 260 m and draft ≤ 10 m) may enter only if their capacity is ≤ 1000 TEU.
- Ships with a small length (≤ 260 m), medium draft > 10 and ≤ 12 m, and capacity ≤ 4000 TEU, may enter only if cargo residuals have ≤ 0.75 mg dry weight per cm^2.
- Medium-sized ships (with length > 260 m and < 320 m, and draft > 10 m and ≤ 13 m), and with a cargo capacity < 6000 TEU, may enter only if their residuals have ≤ 0.5 mg dry weight per cm^2.
- Big ships with length between 320 m and 400 m, draft > 13 m, and capacity > 4000 TEU, may enter only if their carried residuals have ≤ 0.25 mg dry weight per cm^2.

Clearance Rules in DMN S-FEEL (Old Version)

Vessel Clearance						
C U	***Cer. Exp.*** (date)	***Length*** (m)	***Draft*** (m)	***Capacity*** (TEU)	***Cargo*** (mg/cm^2)	***Enter***
	≥ 0	≥ 0	≥ 0	≥ 0	≥ 0	**Y,N**
1	$\leq$ `today`	–	–	–	–	**N**
2	$>$ `today`	<260	<10	<1000	–	**Y**
3	$>$ `today`	<260	<10	≥1000	–	**N**
4	$>$ `today`	<260	[10,12]	<4000	≤0.75	**Y**
5	$>$ `today`	<260	[10,12]	<4000	>0.75	**N**
6	$>$ `today`	[260,320)	(10,13]	<6000	≤0.5	**Y**
7	$>$ `today`	[260,320)	(10,13]	<6000	>0.5	**N**
8	$>$ `today`	[320,400)	≥13	>4000	≤0.25	**Y**
9	$>$ `today`	[320,400)	≥13	>4000	>0.25	**N**

Clearance Rules in DMN S-FEEL (Old Version)

Vessel Clearance						
C U	***Cer. Exp.*** (date)	***Length*** (m)	***Draft*** (m)	***Capacity*** (TEU)	***Cargo*** (mg/cm^2)	***Enter***
	≥ 0	≥ 0	≥ 0	≥ 0	≥ 0	**Y,N**
1	$\leq$ `today`	–	–	–	–	**N**
2	$>$ `today`	<260	<10	<1000	–	**Y**
3	$>$ `today`	<260	<10	≥ 1000	–	**N**
4	$>$ `today`	<260	$[10,12]$	<4000	≤ 0.75	**Y**
5	$>$ `today`	<260	$[10,12]$	<4000	>0.75	**N**
6	$>$ `today`	$[260,320)$	$(10,13]$	<6000	≤ 0.5	**Y**
7	$>$ `today`	$[260,320)$	$(10,13]$	<6000	>0.5	**N**
8	$>$ `today`	$[320,400)$	≥ 13	>4000	≤ 0.25	**Y**
9	$>$ `today`	$[320,400)$	≥ 13	>4000	>0.25	**N**

Key Questions

- Is the hit indicator correct?
- Is the table complete?
- Do we need all the input data for a ship to apply the decision?

Clearance Rules in DMN S-FEEL (Old Version)

Vessel Clearance						
C U	***Cer. Exp.*** (date)	***Length*** (m)	***Draft*** (m)	***Capacity*** (TEU)	***Cargo*** (mg/cm^2)	***Enter***
	≥ 0	≥ 0	≥ 0	≥ 0	≥ 0	**Y,N**
1	$\leq$ `today`	–	–	–	–	**N**
2	$>$ `today`	<260	<10	<1000	–	**Y**
3	$>$ `today`	<260	<10	≥ 1000	–	**N**
4	$>$ `today`	<260	$[10,12]$	<4000	≤ 0.75	**Y**
5	$>$ `today`	<260	$[10,12]$	<4000	>0.75	**N**
6	$>$ `today`	$[260,320)$	$(10,13]$	<6000	≤ 0.5	**Y**
7	$>$ `today`	$[260,320)$	$(10,13]$	<6000	>0.5	**N**
8	$>$ `today`	$[320,400)$	≥ 13	>4000	≤ 0.25	**Y**
9	$>$ `today`	$[320,400)$	≥ 13	>4000	>0.25	**N**

Hit indicator

Unique hit: **yes**!

Completeness

- **no** if table considered *in isolation*;
- **yes** if understood *in the context* of the **ship KB.**

Clearance Rules in DMN S-FEEL (Old Version)

Vessel Clearance						
C U	*Cer. Exp.* (date)	*Length* (m)	*Draft* (m)	*Capacity* (TEU)	*Cargo* (mg/cm^2)	*Enter*
	≥ 0	≥ 0	≥ 0	≥ 0	≥ 0	**Y,N**
1	$\leq$ `today`	–	–	–	–	**N**
2	$>$ `today`	<260	<10	<1000	–	**Y**
3	$>$ `today`	<260	<10	≥ 1000	–	**N**
4	$>$ `today`	<260	[10,12]	<4000	≤ 0.75	**Y**
5	$>$ `today`	<260	[10,12]	<4000	>0.75	**N**
6	$>$ `today`	[260,320)	(10,13]	<6000	≤ 0.5	**Y**
7	$>$ `today`	[260,320)	(10,13]	<6000	>0.5	**N**
8	$>$ `today`	[320,400)	≥ 13	>4000	≤ 0.25	**Y**
9	$>$ `today`	[320,400)	≥ 13	>4000	>0.25	**N**

Do we need all physical characteristics of a ship for clearance?

- From **ship type**, using the ship KB one can infer partial information about length, draft and capacity.
- Combined with **certificate expiration** and **cargo residuals**, this is enough to unambiguously apply the decision table!

Sources of Decision Knowledge

- **S-FEEL DMN Decisions.** Defined by the standard.
- **Knowledge Base.** Multi-sorted FOL theory $\mathsf{FOL}(\mathfrak{D})$.
  - Quantification domain: objects Δ + data values from different sorts $\mathfrak{D}$ capturing S-FEEL data types (with comparison predicates).
  - Class: unary predicate interpreted over Δ.
  - Role: Binary predicate relating pairs of objects from Δ.
  - Feature: Binary predicate relating objects from Δ to data values from a selected data type in $\mathfrak{D}$.

  Closed formulae interpreted as constraints.

Sources of Decision Knowledge

- **S-FEEL DMN Decisions.** Defined by the standard.
- **Knowledge Base.** Multi-sorted FOL theory FOL($\mathfrak{D}$).
  - Quantification domain: objects Δ + data values from different sorts $\mathfrak{D}$ capturing S-FEEL data types (with comparison predicates).
  - Class: unary predicate interpreted over Δ.
  - Role: Binary predicate relating pairs of objects from Δ.
  - Feature: Binary predicate relating objects from Δ to data values from a selected data type in $\mathfrak{D}$.

  Closed formulae interpreted as constraints.

Example

Ship Type	**Short**	**Length** (m)	**Draft** (m)	**Capacity** (TEU)
...	*CCV*	135	0 – 9	500

$$\forall s.\mathit{CCV}(s) \rightarrow \mathit{Ship}(s) \wedge \forall l.(\mathit{length}(s,l) \rightarrow l = 135) \wedge \forall d.(\mathit{draft}(s,d) \rightarrow d \geq 0 \wedge d \leq 9) \wedge \forall c.(\mathit{capacity}(s,c) \rightarrow c = 500)$$

Combining Decisions and KBs in 3 Steps

Step 1. Decision tables apply to objects of some class

Identification of the “bridge” class that is subject at once to the constraints of the KB and the decision logic.

Combining Decisions and KBs in 3 Steps

Step 1. Decision tables apply to objects of some class

Identification of the "bridge" class that is subject at once to the constraints of the KB and the decision logic.

Example

Ship is the bridge class linking the Ship KB to the Vessel Clearance decision table.

Combining Decisions and KBs in 3 Steps

Step 2. Decision tables enrich the vocabulary of the KB

Table inputs/outputs denote features of the bridge class:

- Each input I becomes an input feature I.
  - If already used in the KB: type compatibility.
- Each output O and rule r becomes an output feature O_r.
  - A new feature, not already used in the KB.
  - Retains rule provenance (useful in case of multiple hits).

	$\mathbf{I_1}$ (D^i_1)	$\mathbf{I_2}$ (D^i_2)	$\mathbf{I_3}$ (D^i_3)	$\mathbf{O_1}$ (D^o_1)	$\mathbf{O_2}$ (D^o_2)
1					
...					
k					

+

C

...

=

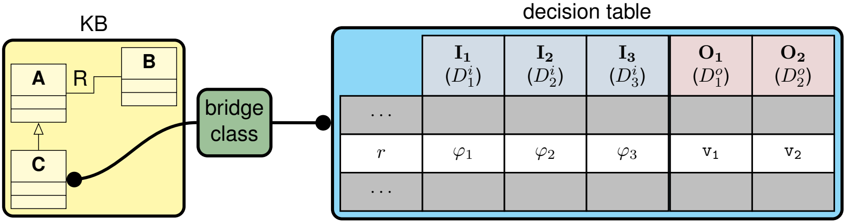
C

...
$I_1 : D^i_1$
$I_2 : D^i_2$
$I_3 : D^i_3$
...
$O_{1,1} : D^o_1$
...
$O_{1,k} : D^o_1$
$O_{2,1} : D^o_2$
...
$O_{2,k} : D^o_2$

Combining Decisions and KBs in 3 Steps

Step 3: combined reasoning

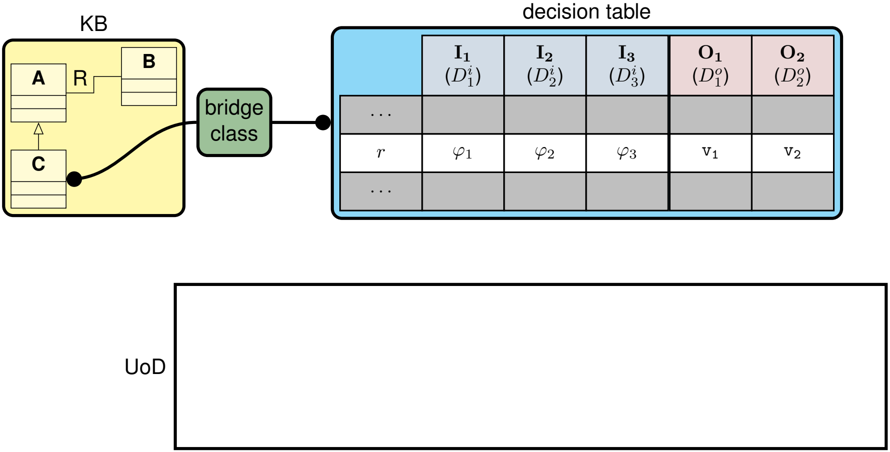
- KB: constrains (some) of the table input features.
- Decision: relates constrained input features to output features.



Combining Decisions and KBs in 3 Steps

Step 3: combined reasoning

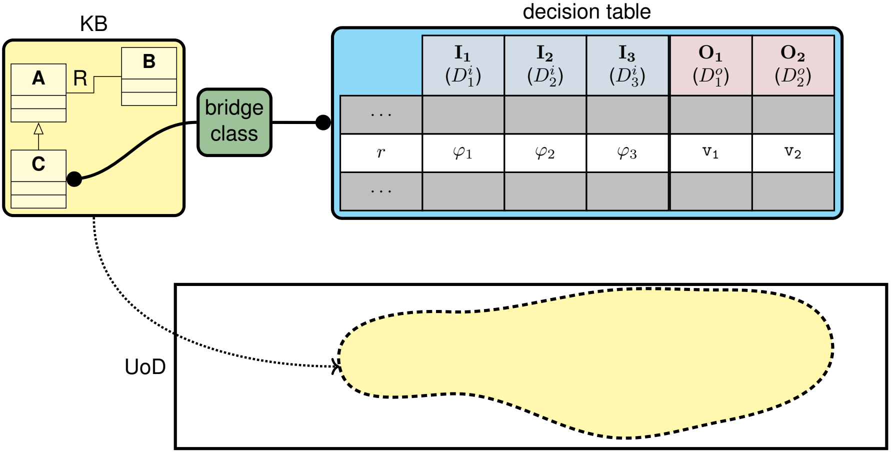
- KB: constrains (some) of the table input features.
- Decision: relates constrained input features to output features.



Combining Decisions and KBs in 3 Steps

Step 3: combined reasoning

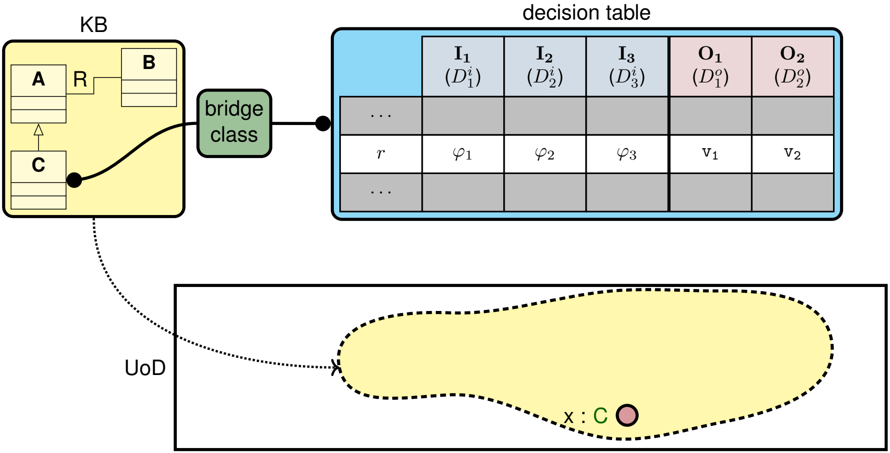
- KB: constrains (some) of the table input features.
- Decision: relates constrained input features to output features.



Combining Decisions and KBs in 3 Steps

Step 3: combined reasoning

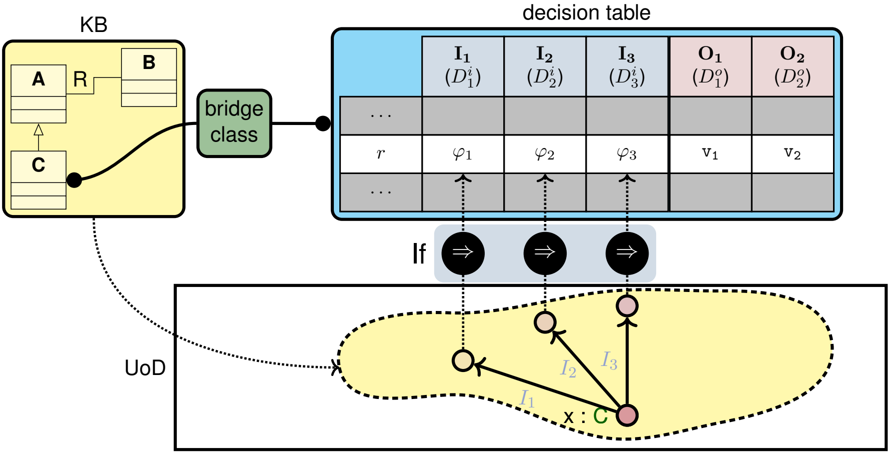
- KB: constrains (some) of the table input features.
- Decision: relates constrained input features to output features.



Combining Decisions and KBs in 3 Steps

Step 3: combined reasoning

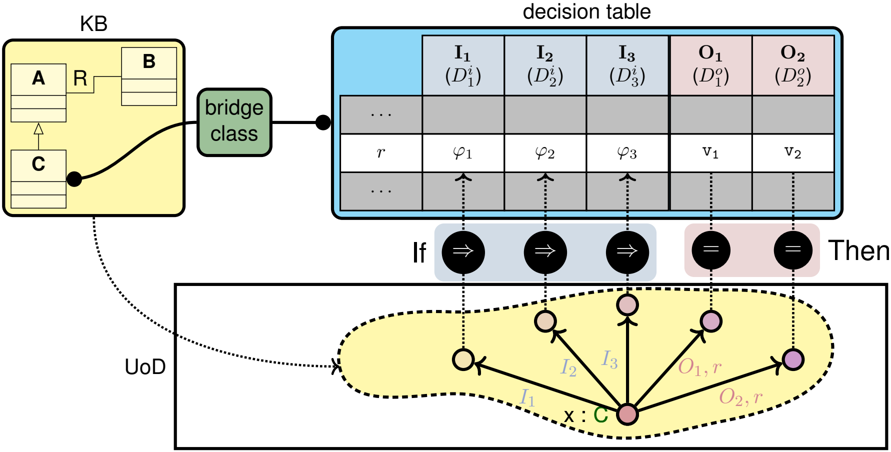
- KB: constrains (some) of the table input features.
- Decision: relates constrained input features to output features.



Combining Decisions and KBs in 3 Steps

Step 3: combined reasoning

- KB: constrains (some) of the table input features.
- Decision: relates constrained input features to output features.



Ships Strike Back

Vessel Clearance					
Cer.Exp. (date)	**Length** (m)	**Draft** (m)	**Capacity** (TEU)	**Cargo** (mg/cm^2)	**Enter** `Y, N`
9 rules					

Ships Strike Back

Vessel Clearance					
Cer.Exp. Real	**Length** Real	**Draft** Real	**Capacity** Real	**Cargo** Real	**Enter** Bool
9 rules					

Ships Strike Back

Vessel Clearance

Cer.Exp. Real	Length Real	Draft Real	Capacity Real	Cargo Real	Enter Bool
9 rules					

+

Ship

Length : Real
Draft : Real
Capacity : Real

Ships Strike Back

Vessel Clearance					
Cer.Exp. Real	**Length** Real	**Draft** Real	**Capacity** Real	**Cargo** Real	**Enter** Bool
9 rules					

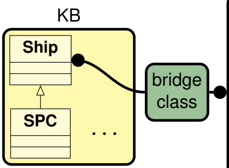
\+

Ship
Length : Real *Draft* : Real *Capacity* : Real

=

Ship
Length : Real *Draft* : Real *Capacity* : Real *Cer.Exp.* : Real *Cargo* : Real $Enter_1$: Bool ... $Enter_9$: Bool

An Empty Panamax Ship Approaches the Harbor...

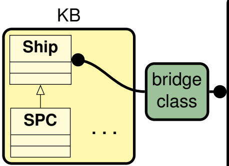


decision table

Vessel Clearance

	Cer.Exp. Real	Length Real	Draft Real	Capacity Real	Cargo Real	Enter Bool
...						
4	$>$ today	< 260	$[10, 12]$	< 4000	≤ 0.75	**Y**
...						

An Empty Panamax Ship Approaches the Harbor. . .



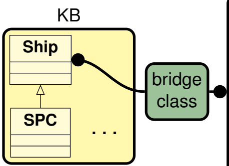
decision table

Vessel Clearance

	Cer.Exp. Real	Length Real	Draft Real	Capacity Real	Cargo Real	Enter Bool
...						
4	$>$ today	< 260	$[10, 12]$	< 4000	≤ 0.75	**Y**
...						

UoD

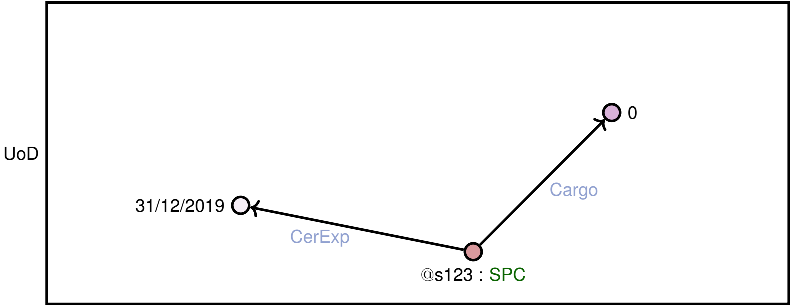
An Empty Panamax Ship Approaches the Harbor…



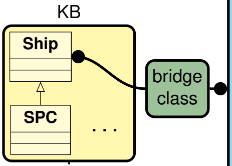
decision table

Vessel Clearance

	Cer.Exp. Real	Length Real	Draft Real	Capacity Real	Cargo Real	Enter Bool
...						
4	$>$ today	< 260	$[10, 12]$	< 4000	≤ 0.75	**Y**
...						

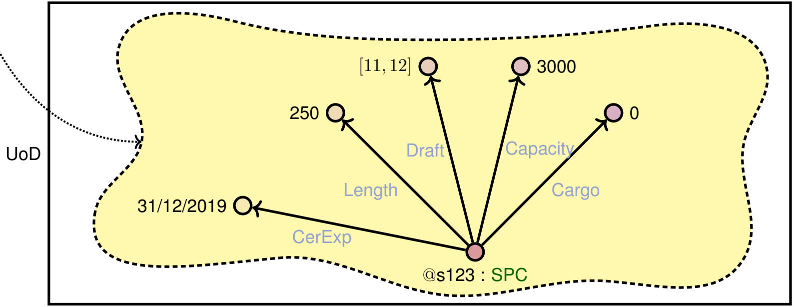


An Empty Panamax Ship Approaches the Harbor. . .

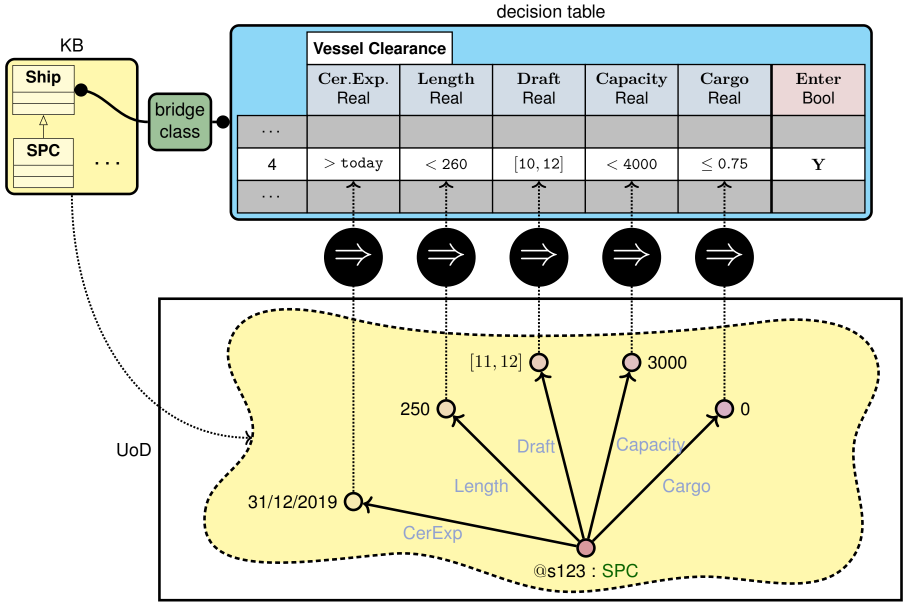


decision table

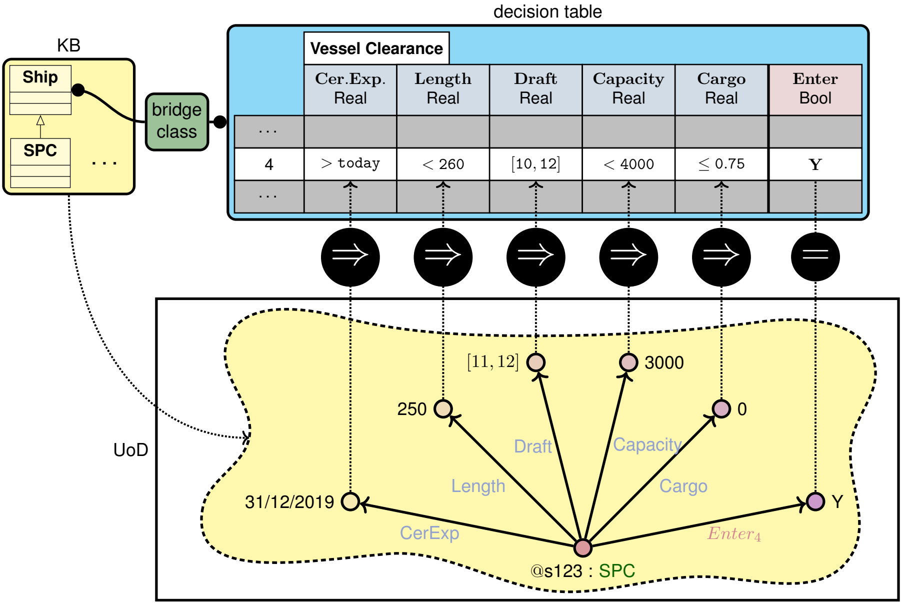
Vessel Clearance	Cer.Exp. Real	Length Real	Draft Real	Capacity Real	Cargo Real	Enter Bool
...						
4	$>$ `today`	< 260	$[10, 12]$	< 4000	≤ 0.75	**Y**
...						



An Empty Panamax Ship Approaches the Harbor. . .



An Empty Panamax Ship Approaches the Harbor. . .



Decision Knowledge Bases

Definition (DKB)

A decision knowledge base over datatypes $\mathfrak{D}$ ($\mathfrak{D}$-DKB, or DKB for short) is a tuple $\langle \Sigma, \mathcal{T}, \mathcal{M}, C, A \rangle$, where:

- $\mathcal{T}$ is a FOL$(\mathfrak{D})$ intensional KB with signature Σ.
- $\mathcal{M}$ is a DMN decision that satisfies the following two typing conditions:
  (output uniqueness) no output attribute of $\mathcal{M}$ is part of Σ;
  (input type compatibility) for every binary predicate $P \in \Sigma$ whose name coincides with an input attribute of $\mathcal{M}$, their types coincide.
- $C \in \Sigma_C$ is the *bridge class*.
- A is an ABox over the extended signature $\Sigma \cup \mathcal{M}.I$.

Decision Knowledge Bases

Definition (DKB)

A decision knowledge base over datatypes $\mathfrak{D}$ ($\mathfrak{D}$-DKB, or DKB for short) is a tuple $\langle \Sigma, \mathcal{T}, \mathcal{M}, C, A \rangle$, where:

- $\mathcal{T}$ is a FOL$(\mathfrak{D})$ intensional KB with signature Σ.
- $\mathcal{M}$ is a DMN decision that satisfies the following two typing conditions:
  (output uniqueness) no output attribute of $\mathcal{M}$ is part of Σ;
  (input type compatibility) for every binary predicate $P \in \Sigma$ whose name coincides with an input attribute of $\mathcal{M}$, their types coincide.
- $C \in \Sigma_C$ is the *bridge class*.
- A is an ABox over the extended signature $\Sigma \cup \mathcal{M}.I$.

Input/output Configuration

Input/output configurations for $\mathcal{M}$ are now simply set of facts over an object of type C.

Reasoning tasks: Compatibility with Hit Indicators

Compatibility with Unique Hit

Input: DKB $\mathcal{X} = \langle \Sigma, \mathcal{T}, \mathcal{M}, C, \emptyset \rangle$ (intensional, no data).
Question: Do rules in $\mathcal{M}$ overlap?

Compatibility with Any Hit

Input: DKB $\mathcal{X} = \langle \Sigma, \mathcal{T}, \mathcal{M}, C, \emptyset \rangle$ (intensional, no data).
Question: Do rules in $\mathcal{M}$ that produce different outputs overlap?

Compatibility with Priority Hit

Input: DKB $\mathcal{X} = \langle \Sigma, \mathcal{T}, \mathcal{M}, C, \emptyset \rangle$ (intensional, no data).
Question: Are there rules in $\mathcal{M}$ masked by others?

Table completeness

Input: DKB $\mathcal{X} = \langle \Sigma, \mathcal{T}, \mathcal{M}, C, \emptyset \rangle$ (intensional, no data).
Question: Does every possible input configuration match a rule in $\mathcal{M}$?

Reasoning tasks: I/O Behavior

I/O Relationship

Input:

- DKB $\mathcal{X} = \langle \Sigma, \mathcal{T}, \mathcal{M}, C, A \rangle$,
- object $\mathtt{c} \in \Delta$ of type C,
- output attribute $\mathbf{b}$ of $\mathcal{M}$,
- value $\mathtt{v}$ with type that of $\mathbf{b}$.

Question: Is it the case that $\mathcal{X}$ assigns $\mathtt{v}$ to $\mathtt{c}$ for attribute $\mathbf{b}$?

Reasoning tasks: I/O Behavior

Output coverage

Input:

- DKB $\mathcal{X} = \langle \Sigma, \mathcal{T}, \mathcal{M}, C, \emptyset \rangle$ (intensional, no data),
- output attribute b of $\mathcal{M}$,
- value v with type that of b.

Question: Is there an input configuration that leads to assign v to b?

Output determinability

Input:

- DKB $\mathcal{X} = \langle \Sigma, \mathcal{T}, \mathcal{M}, C, \emptyset \rangle$ (intensional, no data),
- unary formula $\varphi(x)$ characterising an input template.

Question: Does $\mathcal{M}$ assign an output to each object of type C that satisfies the template formula $\varphi(x)$?

Reasoning tasks: I/O Behavior

Output coverage

Input:

- DKB $\mathcal{X} = \langle \Sigma, \mathcal{T}, \mathcal{M}, C, \emptyset \rangle$ (intensional, no data),
- output attribute $\mathbf{b}$ of $\mathcal{M}$,
- value v with type that of $\mathbf{b}$.

Question: Is there an input configuration that leads to assign v to $\mathbf{b}$?

Output determinability

Input:

- DKB $\mathcal{X} = \langle \Sigma, \mathcal{T}, \mathcal{M}, C, \emptyset \rangle$ (intensional, no data),
- unary formula $\varphi(x)$ characterising an input template.

Question: Does $\mathcal{M}$ assign an output to each object of type C that satisfies the template formula $\varphi(x)$?

Disclaimer

In [___,TPLP2019] we also consider Decision Requirement Graphs and further reasoning tasks.

How to Reason?

Question

Is a DKB different from a conventional KB?

How to Reason?

Question

Is a DKB different from a conventional KB?

Observation

Decision table = a set of additional constraints over the bridge class.

How to Reason?

Question

Is a DKB different from a conventional KB?

Observation

Decision table = a set of additional constraints over the bridge class.

From a DKB to a KB

Given a DKB $\langle \Sigma, \mathcal{T}, \mathcal{M}, C, A \rangle$, construct a conventional KB as follows:

1. Take $\mathcal{T}$ as the initial KB.
2. Encode the attributes of $\mathcal{M}$:
   a. Expand the vocabulary Σ of $\mathcal{T}$ with input/output features from $\mathcal{M}$.
   b. Generate typing and facet constraints for such features.
3. Encode the rules of $\mathcal{M}$: each rule becomes a constraint.

How to Reason?

Question

Is a DKB different from a conventional KB?

Observation

Decision table = a set of additional constraints over the bridge class.

From a DKB to a KB

Given a DKB $\langle \Sigma, \mathcal{T}, \mathcal{M}, C, A \rangle$, construct a conventional KB as follows:

1. Take $\mathcal{T}$ as the initial KB.
2. Encode the attributes of $\mathcal{M}$:
   a. Expand the vocabulary Σ of $\mathcal{T}$ with input/output features from $\mathcal{M}$.
   b. Generate typing and facet constraints for such features.
3. Encode the rules of $\mathcal{M}$: each rule becomes a constraint.

Goal

Reasoning over DKBs as standard reasoning over KBs.

Encoding of Attributes (1)

Extending the signature

- A feature for each input attribute of the decision that is not already used in the KB.
- A feature for each combination of output attribute-rule: output feature + its provenance.

Example

Vessel Clearance

Cer.Exp.	Length	Draft	Capacity	Cargo	Enter
Real	Real	Real	Real	Real	Bool

- Attributes *Length*, *Draft*, *Capacity* correspond to compatible facets in the background KB;
- 2 new features for *CerExp* and *Cargo*;
- 9 new features for *Enter*, i.e., *Enter*$_i$ for rule i ($i \in \{1, \ldots, 9\}$).

Encoding of Attributes (2)

Constraining the features

For each input/output feature, add:

- Typing constraint: the domain of the feature is the bridge concept.
- Functionality constraint: no two attributes of the same kind.
  - For input features: non-ambiguous application of rules.
  - For output features: simply asserts that an output cell contains a single value.

Example

Length
Real

$\longrightarrow$

$$\forall x, y.\textit{length}(x, y) \rightarrow \textit{Ship}(x)$$
$$\forall x, y, z.\textit{length}(x, y) \wedge \textit{length}(x, z) \rightarrow y = z$$

Encoding of S-FEEL Conditions

An S-FEEL condition is a compact representation of unary FOL($\mathfrak{D}$) formula applied to data values.

S-FEEL Translation Function

Given an S-FEEL condition Q, function $\tau^x(Q)$ builds a unary FOL($\mathfrak{D}$) formula that encodes the application of Q to x.

$$\tau^x(Q) \triangleq \begin{cases} \mathit{true} & \text{if } \mathcal{Q} = \text{``}-\text{''} \\ x \neq \mathtt{v} & \text{if } \mathcal{Q} = \text{``}\mathtt{not(v)}\text{''} \\ x = \mathtt{v} & \text{if } \mathcal{Q} = \text{``}\mathtt{v}\text{''} \\ x \approx \mathtt{v} & \text{if } \mathcal{Q} = \text{``}\approx \mathtt{v}\text{'' and } \approx \in \{<, >, \leq, \geq\} \\ x > \mathtt{v}_1 \wedge x < \mathtt{v}_2 & \text{if } \mathcal{Q} = \text{``}(\mathtt{v}_1..\mathtt{v}_2)\text{''} \\ \ldots & \text{(similarly for the other types of intervals)} \\ \tau^x(\mathcal{Q}_1) \vee \tau^x(\mathcal{Q}_2) & \text{if } \mathcal{Q} = \text{``}\mathcal{Q}_1,\mathcal{Q}_2\text{''} \end{cases}$$

Encoding of Attribute Facets

Restrict the acceptable values

For each input/output feature, add:

- Facet constraint: restricts the acceptable values of the feature range.
  - The facet is an S-FEEL condition: just translate it to get the constraint.

Example

Length Real
≥ 0

$\longrightarrow$ $\forall x, y.\textit{length}(x, y) \rightarrow \tau^y('> 0')$

Encoding of Attribute Facets

Restrict the acceptable values

For each input/output feature, add:

- Facet constraint: restricts the acceptable values of the feature range.
  - The facet is an S-FEEL condition: just translate it to get the constraint.

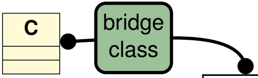
Example

Length Real
≥ 0

$\longrightarrow$ $\forall x, y.\mathit{length}(x, y) \rightarrow y \geq 0$

Encoding of Rules

Rules as logical implications

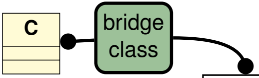


	$\mathbf{I_1}$ (D_1^i)	$\dots$	$\mathbf{I_n}$ (D_n^i)	$\mathbf{O_1}$ (D_1^o)	$\dots$	$\mathbf{O_m}$ (D_m^o)
r	φ_1	$\dots$	φ_n	$\mathtt{v}_1$	$\dots$	$\mathtt{v}_\mathtt{m}$

Encoding of Rules

Rules as logical implications

For every instance of the bridge class:



	$\mathbf{I_1}$ (D_1^i)	$\ldots$	$\mathbf{I_n}$ (D_n^i)	$\mathbf{O_1}$ (D_1^o)	$\ldots$	$\mathbf{O_m}$ (D_m^o)
r	φ_1	$\ldots$	φ_n	$\mathtt{v_1}$	$\ldots$	$\mathtt{v_m}$

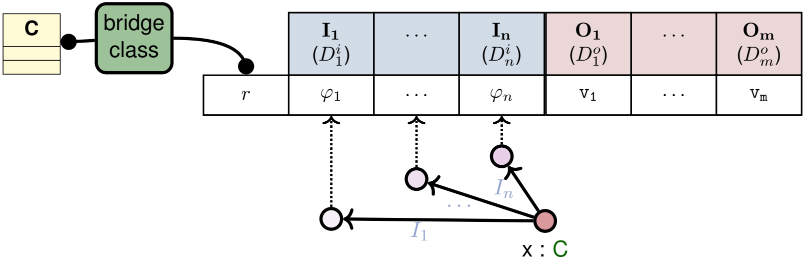
$\forall x.C(x)$

Encoding of Rules

Rules as logical implications

For every instance of the bridge class:

If each input feature satisfies the corresponding input cell condition

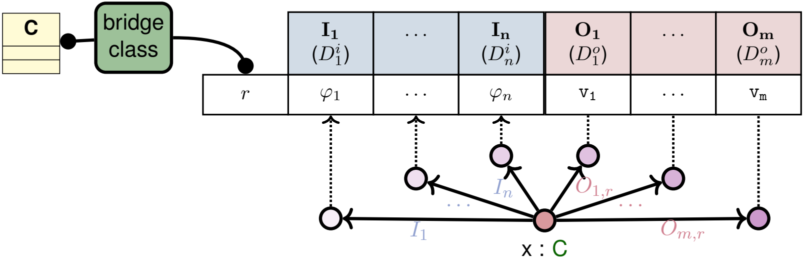


$$\forall x.C(x) \land \forall \vec{y}. \bigwedge_{j \in \{1,\ldots,n\}} (I_j(x, y_j) \land \tau^{y_j}(\varphi_j))$$

Encoding of Rules

Rules as logical implications

For every instance of the bridge class:
If each input feature satisfies the corresponding input cell condition
Then each output feature points to the value in the corresponding output cell



$$\forall x.C(x) \wedge \forall \vec{y}. \bigwedge_{j \in \{1,\ldots,n\}} (I_j(x, y_j) \wedge \tau^{y_j}(\varphi_j)) \rightarrow \exists \vec{z}. \bigwedge_{k \in \{1,\ldots,m\}} (O_{k,r}(x, z) \wedge z = \mathtt{v_k})$$

Encoding of Rules

Example

	Vessel Clearance					
	Cer.Exp. Real	**Length** Real	**Draft** Real	**Capacity** Real	**Cargo** Real	**Enter** Bool
2	$>$ `today`	< 260	< 10	< 1000	$-$	`Y`

Encoding of rule #2

$$\forall x, e, l, d, c.\mathit{exp}(x, e) \land e > \texttt{today} \land \mathit{length}(x, l) \land l < \texttt{260} \\ \land \mathit{draft}(x, d) \land d < \texttt{10} \land \mathit{cap}(x, c) \land c < \texttt{1000} \rightarrow \exists o.\mathit{enter}_2(x, o) \\ \land o = \texttt{Y}.$$

Reasoning over DKBs as Standard Reasoning over KBs

Fact

All DKB reasoning tasks can be turned into **logical implication** tests in FOL($\mathfrak{D}$).

Computationally, this is of no help.

Reasoning over DKBs as Standard Reasoning over KBs

Fact

All DKB reasoning tasks can be turned into **logical implication** tests in FOL($\mathfrak{D}$).

Computationally, this is of no help.

Goal

Investigate suitable fragments of FOL($\mathfrak{D}$) that:

- Are expressive enough to encode DMN DRGs + S-FEEL decisions.
- Are computationally feasible (with complexity guarantees).

Reasoning over DKBs as Standard Reasoning over KBs

Fact

All DKB reasoning tasks can be turned into **logical implication** tests in FOL($\mathfrak{D}$).

Computationally, this is of no help.

Goal

Investigate suitable fragments of FOL($\mathfrak{D}$) that:

- Are expressive enough to encode DMN DRGs + S-FEEL decisions.
- Are computationally feasible (with complexity guarantees).

Setting

Description logics with data types are the natural candidate for this.

The $\mathcal{ALCH}(\mathfrak{D})$ Logic [Ortiz et al, AAAI2008;___,TPLP2019]

Main features

- Well-known $\mathcal{ALC}$ + multiple data types that do not interact with each other.
- Reasoning (e.g., subsumption): EXPTIME-complete (like $\mathcal{ALC}$).

$\mathcal{ALCH}(\mathfrak{D})$ DKBs

Decision Knowledge Bases where background knowledge is expressed as an $\mathcal{ALCH}(\mathfrak{D})$ ontology.

The $\mathcal{ALCH}(\mathfrak{D})$ Logic [Ortiz et al, AAAI2008;___,TPLP2019]

Main features

- Well-known $\mathcal{ALC}$ + multiple data types that do not interact with each other.
- Reasoning (e.g., subsumption): EXPTIME-complete (like $\mathcal{ALC}$).

$\mathcal{ALCH}(\mathfrak{D})$ DKBs

Decision Knowledge Bases where background knowledge is expressed as an $\mathcal{ALCH}(\mathfrak{D})$ ontology.

Key Observation

All constraints seen so far can be encoded in $\mathcal{ALCH}(\mathfrak{D})$.
- Each S-FEEL rule becomes a subsumption assertion in $\mathcal{ALCH}(\mathfrak{D})$.

Encoding S-FEEL rules into $\mathcal{ALCH}(\mathfrak{D})$

Example

	Vessel Clearance					
	Cer.Exp. Real	Length Real	Draft Real	Capacity Real	Cargo Real	Enter Bool
2	$>$ `today`	< 260	< 10	< 1000	$-$	Y

Encoding of rule #2 in FOL$(\mathfrak{D})$

$$\forall x, e, l, d, c.\mathit{exp}(x, e) \wedge e > \texttt{today} \wedge \mathit{length}(x, l) \wedge l < 260 \\ \wedge \mathit{draft}(x, d) \wedge d < 10 \wedge \mathit{cap}(x, c) \wedge c < 1000 \rightarrow \exists o.\mathit{enter}_2(x, o) \\ \wedge o = \texttt{Y}.$$

Encoding of rule #2 in $\mathcal{ALCH}(\mathfrak{D})$

$$\forall \mathit{exp}.\mathit{real}[>_{\texttt{today}}] \sqcap \forall \mathit{length}.\mathit{real}[<_{260}] \\ \sqcap \forall \mathit{draft}.\mathit{real}[<_{10}] \sqcap \forall \mathit{cap}.\mathit{real}[<_{1000}] \sqsubseteq \exists \mathit{enter}_2 \sqcap \forall \mathit{enter}_2.\mathit{string}[=_{\texttt{Y}}]$$

Main Results: Complexity

Theorem

Consider an $\mathcal{ALCH}(\mathfrak{D})$ *DKB. The encoding into* $\mathsf{FOL}(\mathfrak{D})$ *is logically equivalent to the encoding into* $\mathcal{ALCH}(\mathfrak{D})$.

Main Results: Complexity

Theorem

Consider an $\mathcal{ALCH}(\mathfrak{D})$ DKB. The encoding into $\mathsf{FOL}(\mathfrak{D})$ *is logically equivalent to the encoding into $\mathcal{ALCH}(\mathfrak{D})$.*

Theorem

All DKBs reasoning tasks can be decided in ExpTime *for $\mathcal{ALCH}(\mathfrak{D})$ DKBs.*

Proof.

Reduction from each reasoning task to a polynomial number of instance or subsumption checks w.r.t. an $\mathcal{ALCH}(\mathfrak{D})$ KB, each of which can be decided in ExpTime. □

Main Results: Complexity

Theorem

Consider an $\mathcal{ALCH}(\mathfrak{D})$ DKB. The encoding into $\mathsf{FOL}(\mathfrak{D})$ *is logically equivalent to the encoding into $\mathcal{ALCH}(\mathfrak{D})$.*

Theorem

All DKBs reasoning tasks can be decided in ExpTime *for $\mathcal{ALCH}(\mathfrak{D})$ DKBs.*

Proof.

Reduction from each reasoning task to a polynomial number of instance or subsumption checks w.r.t. an $\mathcal{ALCH}(\mathfrak{D})$ KB, each of which can be decided in ExpTime. □

UML + S-FEEL DMN = OMG^2

Similar results can be obtained using $\mathcal{ALCQI}$ as the base logic.

- $\mathcal{ALCQI}$ is the DL that captures UML class diagrams.

Main Results: Actual Reasoning

OWL 2 standard reasoners work

- $\mathcal{ALCH}(\mathfrak{D})$ datatypes come with unary predicates only.
- Hence $\mathcal{ALCH}(\mathfrak{D})$ DKBs can be directly represented as OWL 2 ontologies.

Main Results: Actual Reasoning

OWL 2 standard reasoners work

- $\mathcal{ALCH}(\mathfrak{D})$ datatypes come with unary predicates only.
- Hence $\mathcal{ALCH}(\mathfrak{D})$ DKBs can be directly represented as OWL 2 ontologies.

Datatypes fading away

All reasoning tasks over intensional $\mathcal{ALCH}(\mathfrak{D})$ DKBs (no data) can be encoded into standard $\mathcal{ALCH}$ reasoning tasks without datatypes.

- In the compilation process, datatype reasoning is invoked.
- Open whether this gives an improvement over OWL 2 reasoners.

Main Results: Actual Reasoning

OWL 2 standard reasoners work

- $\mathcal{ALCH}(\mathfrak{D})$ datatypes come with unary predicates only.
- Hence $\mathcal{ALCH}(\mathfrak{D})$ DKBs can be directly represented as OWL 2 ontologies.

Datatypes fading away

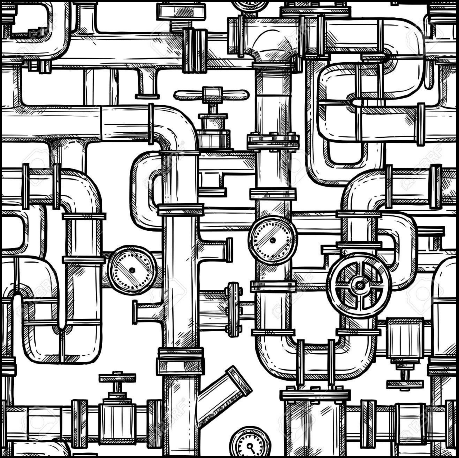
All reasoning tasks over intensional $\mathcal{ALCH}(\mathfrak{D})$ DKBs (no data) can be encoded into standard $\mathcal{ALCH}$ reasoning tasks without datatypes.

- In the compilation process, datatype reasoning is invoked.
- Open whether this gives an improvement over OWL 2 reasoners.

Lightweight DKBs

S-FEEL decisions: expressible in the lightweight DL $\textit{DL-Lite}_{bool}^{(\mathcal{HN})}(\mathfrak{D})$.

- Not enough to capture DRGs.
- Lightweight DLs with datatypes much less investigated than their more expressive companions.

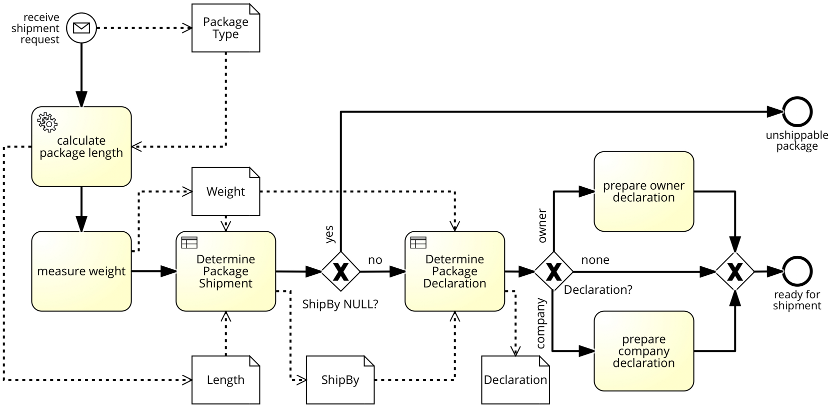


Second Course

Data-aware processes
routing cases based on decisions

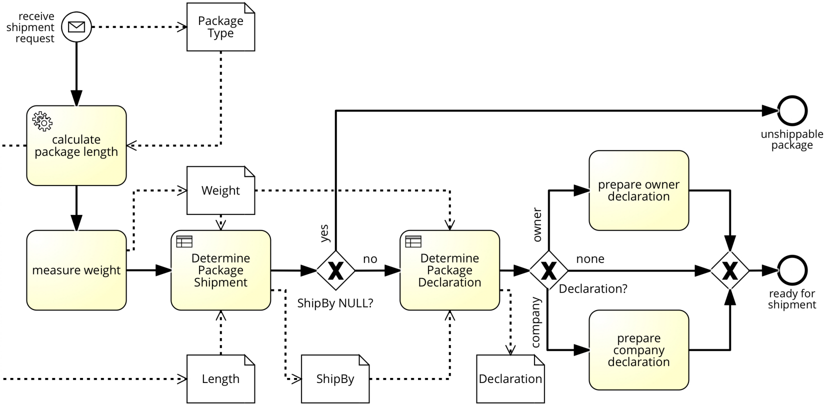
Shipping packages

BLACKSHIP adopts the following BPMN process to ship packages.



Shipping packages

BLACKSHIP adopts the following BPMN process to ship packages.

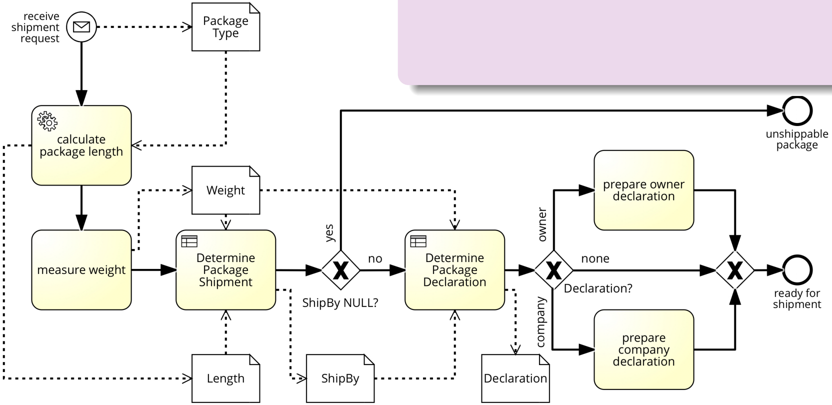


Question

Is the process correct?

The control-flow answer

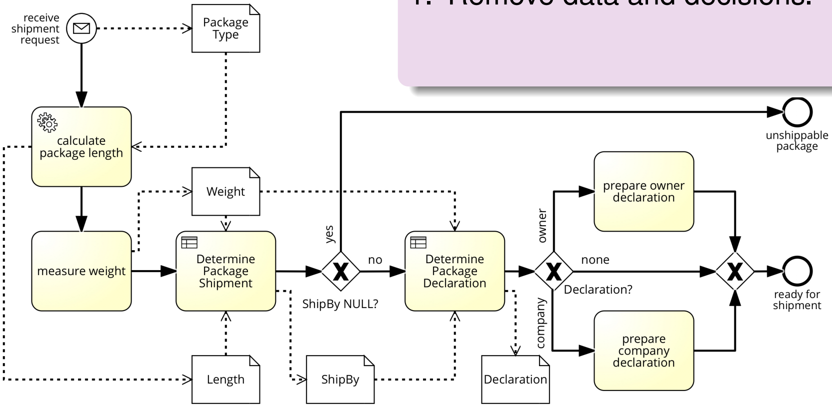
Steps (under case isolation)



The control-flow answer

Steps (under case isolation)

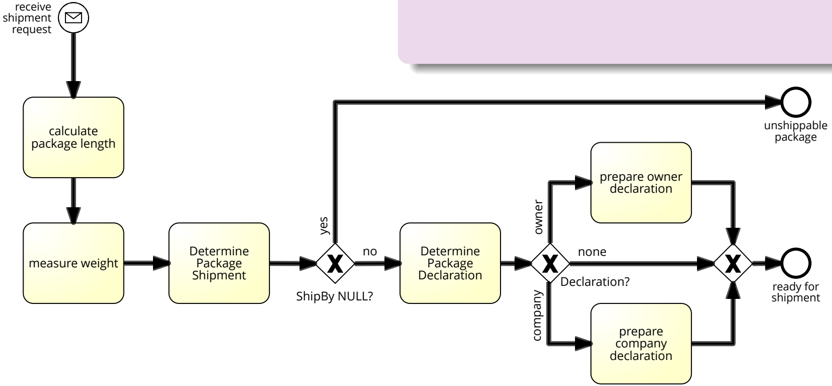
1. Remove data and decisions.



The control-flow answer

Steps (under case isolation)

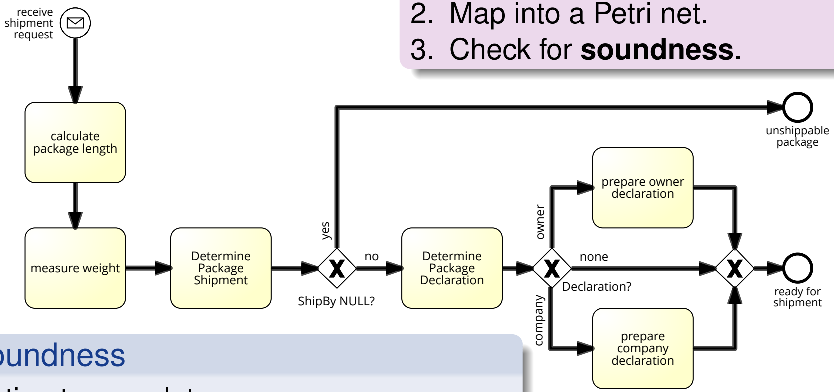
1. Remove data and decisions.



The control-flow answer

Steps (under case isolation)

1. Remove data and decisions.
2. Map into a Petri net.
3. Check for **soundness**.



Soundness

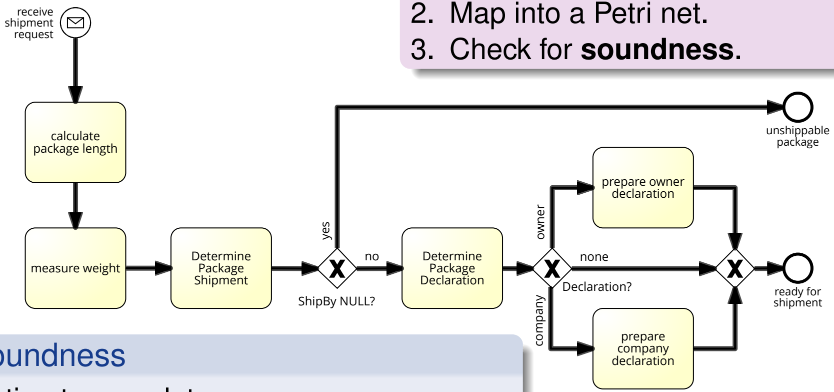
Option to complete:

1. the final marking is always reachable;
2. it is reached always in a ‘clean’ way;
3. there are no dead tasks.

The control-flow answer

Steps (under case isolation)

1. Remove data and decisions.
2. Map into a Petri net.
3. Check for **soundness**.



Soundness

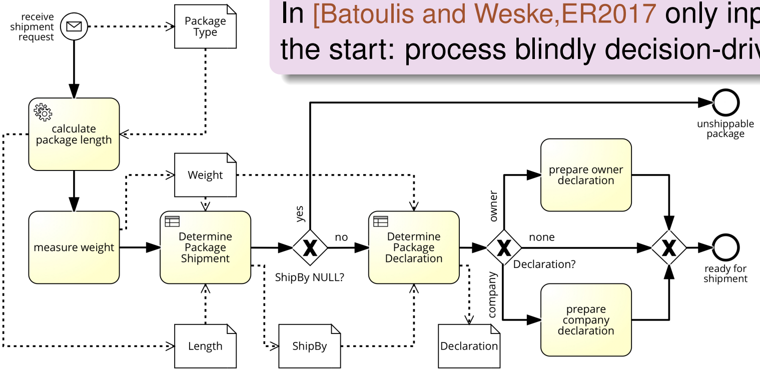
Option to complete:

1. the final marking is always reachable;
2. it is reached always in a ‘clean’ way;
3. there are no dead tasks.

Verdict

Sound!

The real answer



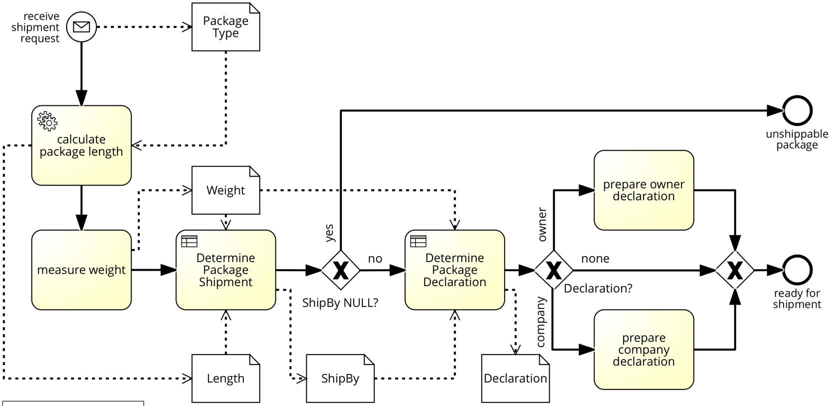
More than decision-aware processes

In [Batoulis and Weske,ER2017] only input at the start: process blindly decision-driven.

Data-awareness brings questions

- Which types for data? Who inputs data? What are the constraints on the inputs?
- What is the decision logic? Which decisions attached to business rule tasks?
- How to lift soundness to **data-aware soundness**?

Back to shipments



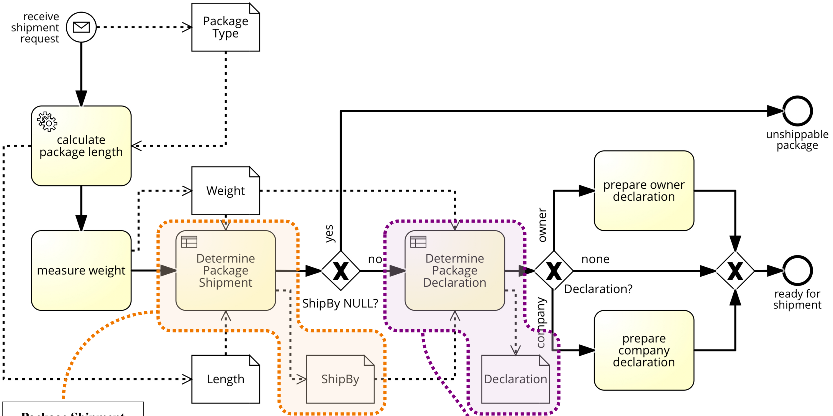
Package Shipment

P	Length (m)	Weight (kg)	ShipBy
	> 0	> 0	`car, truck`
1	(0.0,1.0]	(0, 5]	`car`
2	(0.0,0.6]	(5,10]	`truck`
3	(0.6,1.0]	(4,10]	`truck`
4	(1.0,1.5]	(0, 3]	`car`
5	(1.0,2.0]	(3,10]	`truck`

Package Declaration

U	ShipBy	Weight (kg)	Declaration
	`car,truck`	> 0	<u>`none`</u>, `owner`, `company`
1	`car`	≥ 6	`owner`
2	`truck`	≥ 8	`company`

Back to shipments



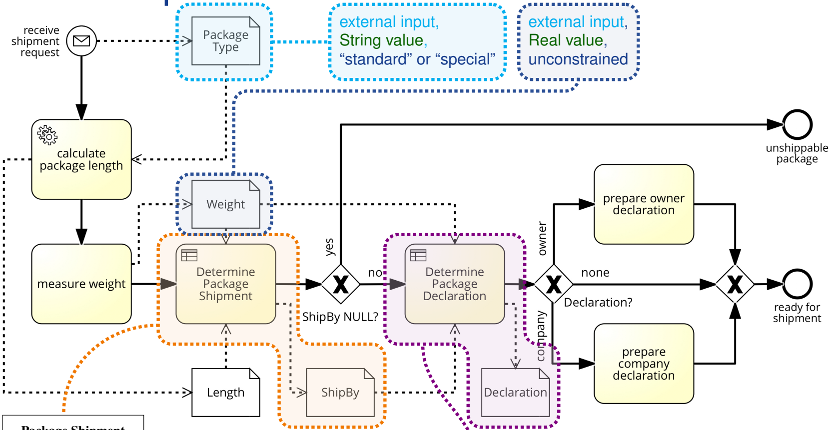
Package Shipment

P	Length (m)	Weight (kg)	ShipBy
	> 0	> 0	`car, truck`
1	(0.0,1.0]	(0, 5]	`car`
2	(0.0,0.6]	(5,10]	`truck`
3	(0.6,1.0]	(4,10]	`truck`
4	(1.0,1.5]	(0, 3]	`car`
5	(1.0,2.0]	(3,10]	`truck`

Package Declaration

U	ShipBy	Weight (kg)	Declaration
	`car,truck`	> 0	`none`, `owner`, `company`
1	`car`	≥ 6	`owner`
2	`truck`	≥ 8	`company`

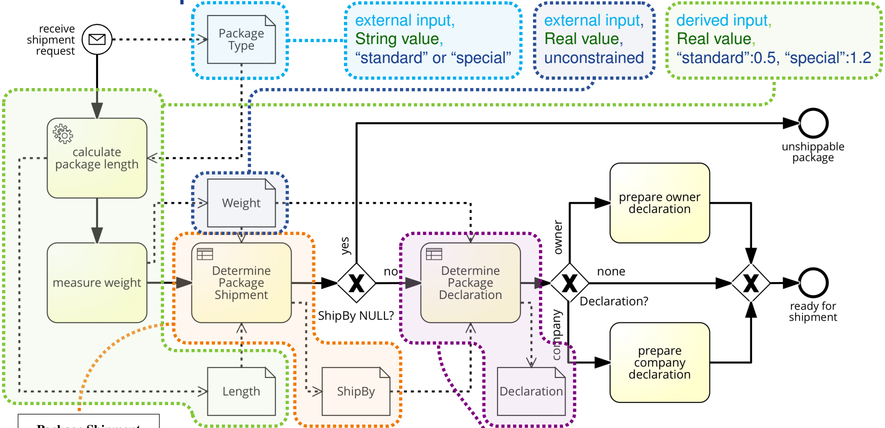
Back to shipments



Package Shipment			
P	Length (m)	Weight (kg)	ShipBy
	> 0	> 0	`car, truck`
1	(0.0,1.0]	(0, 5]	`car`
2	(0.0,0.6]	(5,10]	`truck`
3	(0.6,1.0]	(4,10]	`truck`
4	(1.0,1.5]	(0, 3]	`car`
5	(1.0,2.0]	(3,10]	`truck`

Package Declaration			
U	ShipBy	Weight (kg)	Declaration
	`car,truck`	> 0	`none`, `owner`, `company`
1	`car`	≥ 6	`owner`
2	`truck`	≥ 8	`company`

Back to shipments



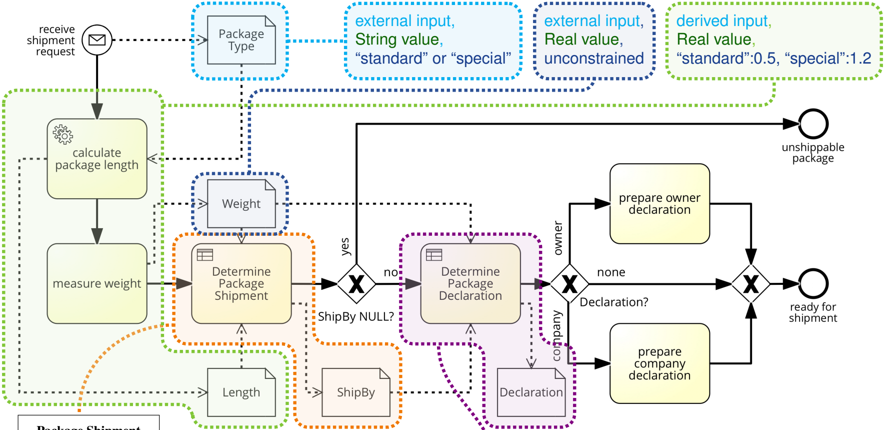
Package Shipment

P	Length (m)	Weight (kg)	ShipBy
	> 0	> 0	car, truck
1	(0.0,1.0]	(0, 5]	car
2	(0.0,0.6]	(5,10]	truck
3	(0.6,1.0]	(4,10]	truck
4	(1.0,1.5]	(0, 3]	car
5	(1.0,2.0]	(3,10]	truck

Package Declaration

U	ShipBy	Weight (kg)	Declaration
	car,truck	> 0	none, owner, company
1	car	≥ 6	owner
2	truck	≥ 8	company

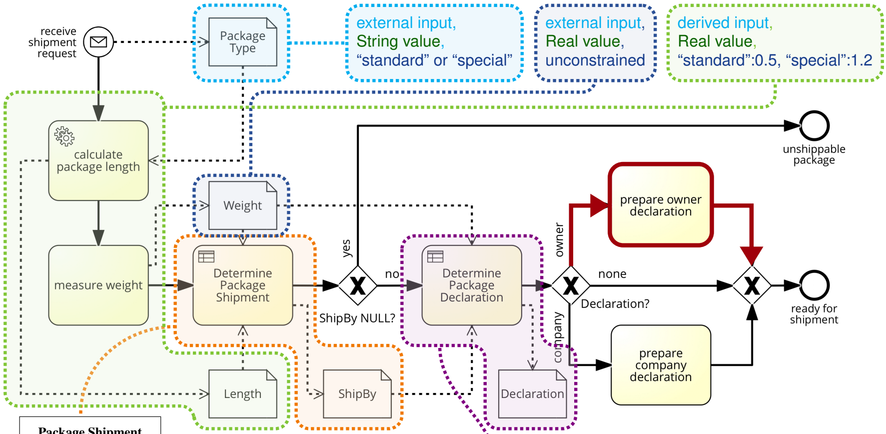
Sound???



Package Shipment			
P	Length (m)	Weight (kg)	ShipBy
	> 0	> 0	`car, truck`
1	(0.0,1.0]	(0, 5]	`car`
2	(0.0,0.6]	(5,10]	`truck`
3	(0.6,1.0]	(4,10]	`truck`
4	(1.0,1.5]	(0, 3]	`car`
5	(1.0,2.0]	(3,10]	`truck`

Package Declaration			
U	ShipBy	Weight (kg)	Declaration
	`car,truck`	> 0	`none`, `owner`, `company`
1	`car`	≥ 6	`owner`
2	`truck`	≥ 8	`company`

Sound??? NO!



Package Shipment

P	Length (m)	Weight (kg)	ShipBy
	> 0	> 0	car, truck
1	(0.0,1.0]	(0, 5]	car
2	(0.0,0.6]	(5,10]	truck
3	(0.6,1.0]	(4,10]	truck
4	(1.0,1.5]	(0, 3]	car
5	(1.0,2.0]	(3,10]	truck

Package Declaration

U	ShipBy	Weight (kg)	Declaration
	car,truck	> 0	<u>none</u>, owner, company
1	car	≥ 6	owner
2	truck	≥ 8	company

In a broader context...

In recent years, there has been an increasing interest in enriching the control-flow perspective of processes with additional dimensions.

The **data perspective** is a prominent one.

In a broader context...

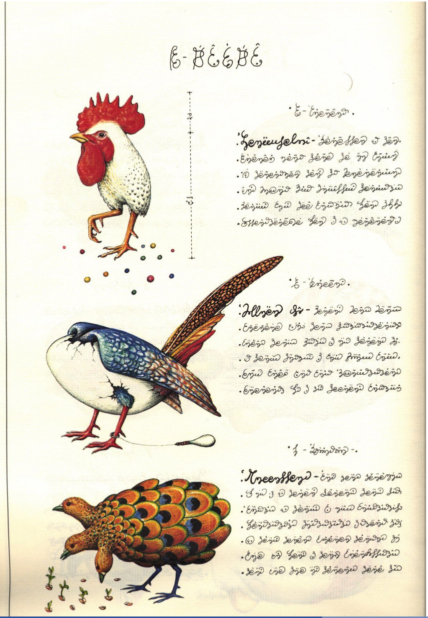
In recent years, there has been an increasing interest in enriching the control-flow perspective of processes with additional dimensions.

The **data perspective** is a prominent one.

Warning

Data range over infinite domains.
$\rightarrow$ Infinitely many process executions in number and length.
$\rightarrow$ Finite-state model checking techniques do not readily apply.

The multifaceted ecosystem of data-aware processes



Control-flow

Petri nets, condition-action rules, declarative constraints, ...

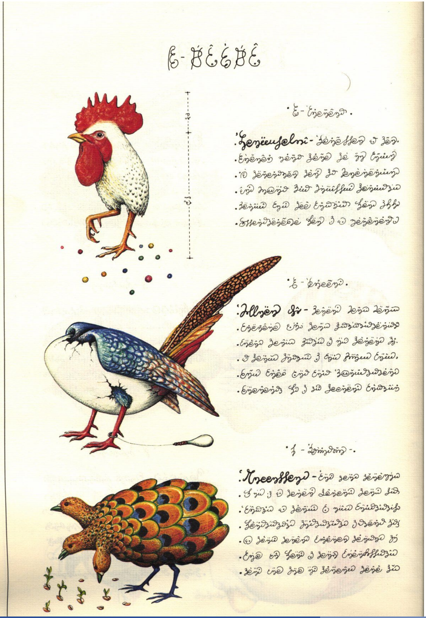
Data

Variables, relational, relational with constraints, semi-stuctured, under incomplete information, ...

Integration

Data access, query, manipulation, external inputs, ...

The multifaceted ecosystem of data-aware processes



Control-flow

Petri nets, condition-action rules, declarative constraints, . . .

Data

Variables, relational, relational with constraints, semi-stuctured, under incomplete information, . . .

Integration

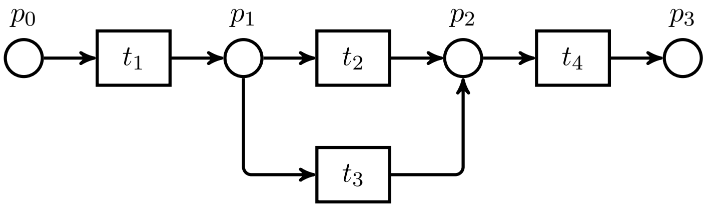
Data access, query, manipulation, external inputs, . . .

Question

Which combination?

Data Petri Nets [Mannhardt,PhD2018;___,ER2018;___,ACSD2019]

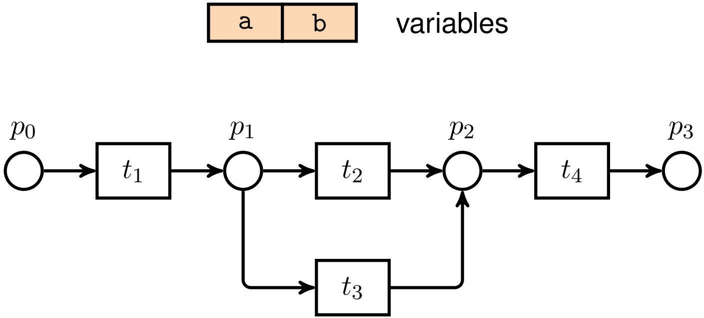
We focus on DPNs, a data-aware extension of P/T nets:



Data Petri Nets [Mannhardt,PhD2018;___,ER2018;___,ACSD2019]

We focus on DPNs, a data-aware extension of P/T nets:

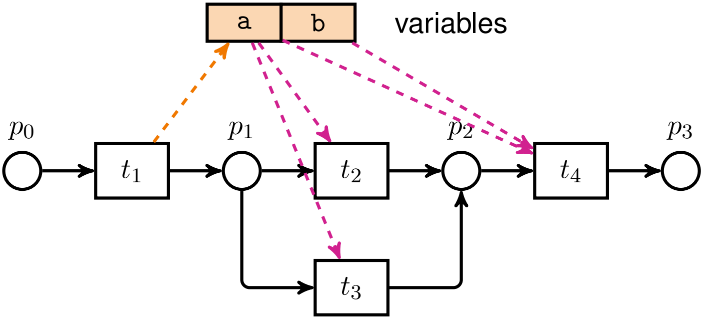
- the net is enriched with a finite set of data variables of different types, with typically **infinite** domain



Data Petri Nets [Mannhardt,PhD2018;___,ER2018;___,ACSD2019]

We focus on DPNs, a data-aware extension of P/T nets:

- the net is enriched with a finite set of data variables of different types, with typically **infinite** domain
- transitions read and update these variables.



Data Petri Nets

- DPNs are less expressive than Petri nets where data are carried by tokens
- but can capture business processes operating over simple case data, taking complex decisions based on these data.

This captures the interesting class of activity-centric business processes that operate over scalar case data, and that use decision models to route the process.

We adopt the richest variant of DPN studied so far [___,ACSD2019].

Data Petri Nets

- DPNs are less expressive than Petri nets where data are carried by tokens
- but can capture business processes operating over simple case data, taking complex decisions based on these data.

This captures the interesting class of activity-centric business processes that operate over scalar case data, and that use decision models to route the process.

We adopt the richest variant of DPN studied so far [___,ACSD2019].

Relevant for process mining too!

DPNs can be discovered from event data [Mannhardt et al,CAiSE2016].
Two-step approach:

1. Discover a Petri net.
2. For each choice point, mine decision tree.

But...

Data Petri Nets

- DPNs are less expressive than Petri nets where data are carried by tokens
- but can capture business processes operating over simple case data, taking complex decisions based on these data.

This captures the interesting class of activity-centric business processes that operate over scalar case data, and that use decision models to route the process.

We adopt the richest variant of DPN studied so far [___,ACSD2019].

Relevant for process mining too!

DPNs can be discovered from event data [Mannhardt et al,CAiSE2016].
Two-step approach:

1. Discover a Petri net.
2. For each choice point, mine decision tree.

But... **No guarantee that the obtained net is sound!**

DPN: formal definition 1/3

Definition (Domain)

Pair $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \Sigma_{\mathcal{D}} \rangle$ where $\Delta_{\mathcal{D}}$ is a set of possible values and $\Sigma_{\mathcal{D}}$ is the set of binary predicates on $\Delta_{\mathcal{D}}$ (closed under negation).

$\mathcal{D}_{\mathbb{R}} = \langle \mathbb{R}, \{<, >, =\} \rangle$
$\mathcal{D}_{\mathbb{Z}} = \langle \mathbb{Z}, \{<, >, =\} \rangle$ (use with care **within loops**!)
$\mathcal{D}_{bool} = \langle \{\mathsf{true}, \mathsf{false}\}, \{=\} \rangle$
$\mathcal{D}_{string} = \langle \mathbb{S}, \{=\} \rangle$

Matches S-FEEL datatypes.

DPN: formal definition 1/3

Definition (Domain)

Pair $\mathcal{D} = \langle \Delta_\mathcal{D}, \Sigma_\mathcal{D} \rangle$ where $\Delta_\mathcal{D}$ is a set of possible values and $\Sigma_\mathcal{D}$ is the set of binary predicates on $\Delta_\mathcal{D}$ (closed under negation).

$\mathcal{D}_\mathbb{R} = \langle \mathbb{R}, \{<, >, =\} \rangle$
$\mathcal{D}_\mathbb{Z} = \langle \mathbb{Z}, \{<, >, =\} \rangle$ (use with care **within loops**!)
$\mathcal{D}_{bool} = \langle \{\mathsf{true}, \mathsf{false}\}, \{=\} \rangle$
$\mathcal{D}_{string} = \langle \mathbb{S}, \{=\} \rangle$

Matches S-FEEL datatypes.

Variables

Typed, and distinguishing read vs write:

$$V^r = \{v^r \mid v \in V\} \qquad V^w = \{v^w \mid v \in V\}$$

DPN: formal definition 2/3

Definition (Guards)

Given a set of typed variables V, the set of possible *guards* $\mathcal{C}_V$ is the largest set containing the following:

- $v_{\mathcal{D}} \odot \Delta_{\mathcal{D}}$ iff $v \in (V^r \cup V^w)$ and $\odot \in \Sigma_{\mathcal{D}}$;
- $v_{1\mathcal{D}} \odot v_{2\mathcal{D}}$ iff $v_1 \in (V^r \cup V^w)$, $v_2 \in V^r$ and $\odot \in \Sigma_{\mathcal{D}}$;

We use constraints to model the *guard* conditions of transitions, for example ($\mathtt{a}, \mathtt{b} \in V$):

- $\mathtt{a}^r > 0$
- $\mathtt{a}^w > 0$
- $\mathtt{a}^r \neq \mathtt{b}^r$
- $\mathtt{a}^w \geq \mathtt{b}^r$

Richer than S-FEEL atomic conditions

Variable-to-variable conditions go beyond S-FEEL:

- processes including richer decision tables;
- processes including S-FEEL decision tables with parameters.

DPN: formal definition 3/3

Definition (Data Petri Net - DPN)

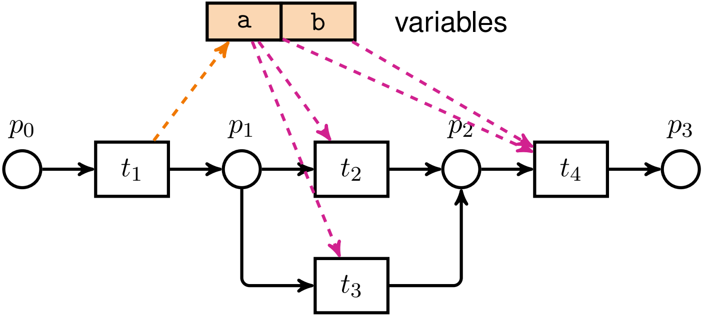
$$\mathcal{N} = \langle P, T, F, V, dom, \alpha_I, read, write, guard \rangle$$

is a Petri net (P, T, F) with additional components, used to describe the additional data perspective of the process model:

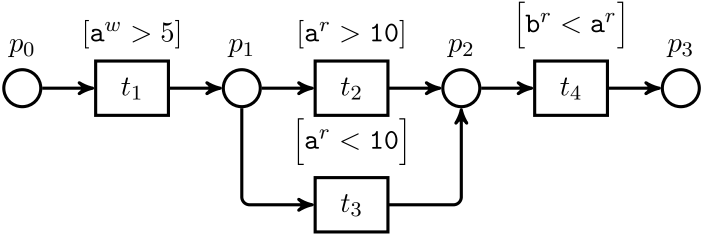
- V is a finite set of process variables;
- dom is a function assigning a domain $\mathcal{D}$ to each $v \in V$;
- α_I is the initial variable assignment;
- $read : T \to 2^V$ returns the set of variable *read* by a transition;
- $write : T \to 2^V$ returns the set of variable *written* by a transition;
- $guard : T \to \Phi(V)$ returns a guard associated with the transition.

We assume an initial marking M_I and a final marking M_F.

DPN: example



DPN: example



- $M_I = \{p_0\}$ and $M_F = \{p_3\}$
- $V = \{\mathtt{a}, \mathtt{b}\}$, both integers
- $\alpha_I(\mathtt{a}) = 0$ and $\alpha_I(\mathtt{b}) = 10$

A couple (M, α) formed by a marking and a variable assignment is called state.

Execution semantics

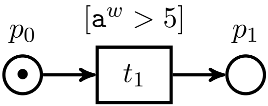
binding of variables in $(V^r \cup V^w)$ to values (in their domain)

Definition (Legal transition firing)

A DPN $\mathcal{N} = \langle P, T, F, V, dom, \alpha_I, read, write, guard \rangle$ evolves from state (M, α) to state (M', α') via transition firing (t, β) with $guard(t) = \phi$ iff:

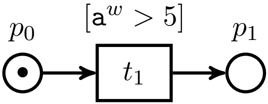
- $\beta(v^r) = \alpha(v)$ if $v \in read(t)$: read variables are not updated;
- the new variable α' is as α but updated as per β:
  $$\alpha'(v) = \begin{cases} \alpha(v) & \text{if } v \notin write(t), \\ \beta(v^w) & \text{otherwise}; \end{cases}$$
- $\phi_{[\beta]} = \mathtt{true}$: the guard is satisfied when we assign value to variables according to β;
- t is enabled: $M(p) > 0$ for every $p \in P$ with $(p, t) \in F$;
- the new marking is computed, denoted $M[t\rangle M'$.

Example of transition firing



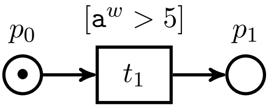
$$(\{p_0\}, \begin{Bmatrix} \alpha(\mathtt{a}) = 0 \\ \alpha(\mathtt{b}) = 10 \end{Bmatrix})$$

Example of transition firing



$$(\{p_0\}, \begin{Bmatrix} \alpha(\mathtt{a}) = 0 \\ \alpha(\mathtt{b}) = 10 \end{Bmatrix}) \xrightarrow{t_1, \beta(\mathtt{a}^w) = 6} (\{p_1\}, \begin{Bmatrix} \alpha(\mathtt{a}) = 6 \\ \alpha(\mathtt{b}) = 10 \end{Bmatrix})$$

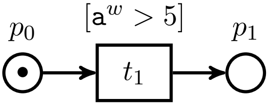
Example of transition firing



$$\left(\{p_0\}, \begin{Bmatrix} \alpha(\texttt{a}) = 0 \\ \alpha(\texttt{b}) = 10 \end{Bmatrix}\right) \xrightarrow{t_1, \beta(\texttt{a}^w) = 6} \left(\{p_1\}, \begin{Bmatrix} \alpha(\texttt{a}) = 6 \\ \alpha(\texttt{b}) = 10 \end{Bmatrix}\right)$$

$$\left(\{p_0\}, \begin{Bmatrix} \alpha(\texttt{a}) = 0 \\ \alpha(\texttt{b}) = 10 \end{Bmatrix}\right) \xrightarrow{t_1, \beta(\texttt{a}^w) = 3} \left(\{p_1\}, \begin{Bmatrix} \alpha(\texttt{a}) = 3 \\ \alpha(\texttt{b}) = 10 \end{Bmatrix}\right)$$

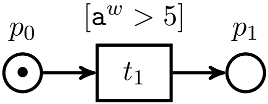
Example of transition firing



$$(\{p_0\}, \begin{Bmatrix} \alpha(\texttt{a}) = 0 \\ \alpha(\texttt{b}) = 10 \end{Bmatrix}) \xrightarrow{t_1, \beta(\texttt{a}^w) = 6} (\{p_1\}, \begin{Bmatrix} \alpha(\texttt{a}) = 6 \\ \alpha(\texttt{b}) = 10 \end{Bmatrix})$$

$$(\{p_0\}, \begin{Bmatrix} \alpha(\texttt{a}) = 0 \\ \alpha(\texttt{b}) = 10 \end{Bmatrix}) \xrightarrow{\cancel{t_1, \beta(\texttt{a}^w) = 3}} (\{p_1\}, \begin{Bmatrix} \alpha(\texttt{a}) = 3 \\ \alpha(\texttt{b}) = 10 \end{Bmatrix})$$

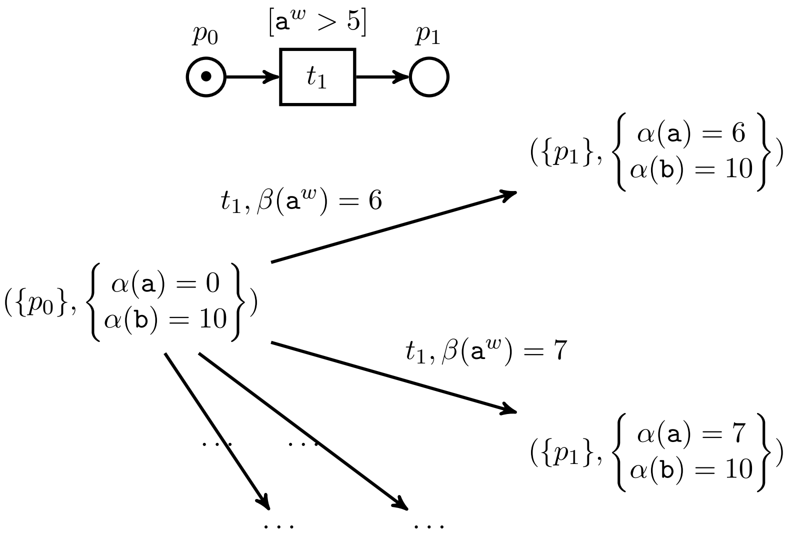
Example of transition firing



$$(\{p_0\}, \begin{Bmatrix} \alpha(\mathtt{a}) = 0 \\ \alpha(\mathtt{b}) = 10 \end{Bmatrix}) \xrightarrow{t_1, \beta(\mathtt{a}^w) = 6} (\{p_1\}, \begin{Bmatrix} \alpha(\mathtt{a}) = 6 \\ \alpha(\mathtt{b}) = 10 \end{Bmatrix})$$

$$(\{p_0\}, \begin{Bmatrix} \alpha(\mathtt{a}) = 0 \\ \alpha(\mathtt{b}) = 10 \end{Bmatrix}) \xrightarrow{t_1, \beta(\mathtt{a}^w) = 7} (\{p_1\}, \begin{Bmatrix} \alpha(\mathtt{a}) = 7 \\ \alpha(\mathtt{b}) = 10 \end{Bmatrix})$$

Example of transition firing



Reachability graph

Definition

The reachability graph of $\mathcal{N}$ is a graph $\langle W, E \rangle$ where:

- $W = Reach_{\mathcal{N}}$ is the set of reachable states of $\mathcal{N}$; and
- $E \subseteq W \times T \times W$ is the set of arcs such that there exists an arc $w \xrightarrow{t,\beta} w'$ iff $w \xrightarrow{t,\beta} w'$ in $\mathcal{N}$.

Reachability graph

Definition

The reachability graph of $\mathcal{N}$ is a graph $\langle W, E \rangle$ where:

- $W = Reach_{\mathcal{N}}$ is the set of reachable states of $\mathcal{N}$; and
- $E \subseteq W \times T \times W$ is the set of arcs such that there exists an arc $w \xrightarrow{t,\beta} w'$ iff $w \xrightarrow{t,\beta} w'$ in $\mathcal{N}$.

Infinite in two dimensions!

- in the length of runs;
- in the branching degree.

Reachability graph

Definition

The reachability graph of $\mathcal{N}$ is a graph $\langle W, E \rangle$ where:

- $W = Reach_{\mathcal{N}}$ is the set of reachable states of $\mathcal{N}$; and
- $E \subseteq W \times T \times W$ is the set of arcs such that there exists an arc $w \xrightarrow{t,\beta} w'$ iff $w \xrightarrow{t,\beta} w'$ in $\mathcal{N}$.

Infinite in two dimensions!

- in the length of runs;
- in the branching degree.

Question

How can we check a suitable data-aware version of classical soundness?

Reachability graph

Definition

The reachability graph of $\mathcal{N}$ is a graph $\langle W, E\rangle$ where:

- $W = Reach_{\mathcal{N}}$ is the set of reachable states of $\mathcal{N}$; and
- $E \subseteq W \times T \times W$ is the set of arcs such that there exists an arc $w \xrightarrow{t,\beta} w'$ iff $w \xrightarrow{t,\beta} w'$ in $\mathcal{N}$.

Infinite in two dimensions!

- in the length of runs;
- in the branching degree.

Question

How can we check a suitable data-aware version of classical soundness?

Answer

Use faithful abstraction!

Data-aware soundness for DPNs

Based on *decision-aware soundness* [Batoulis and Weske,ER2017].

- All the variants studied there can be reconstructed here.

It cannot be defined of the DPN itself, but only on its reachability graph.

Definition (Data-aware soundness - "option to complete")

1: $\forall (M, \alpha) \in Reach_{\mathcal{N}}.\ \ \exists \alpha'.\ (M, \alpha) \xrightarrow{*} (M_F, \alpha')$

2: $\forall (M, \alpha) \in Reach_{\mathcal{N}}.\ \ M \geq M_F \Rightarrow (M = M_F)$

3: $\forall t \in T.\ \exists M_1, M_2, \alpha_1, \alpha_2, \beta.\ (M_1, \alpha_1) \in Reach_{\mathcal{N}}$ and $(M_1, \alpha_1) \xrightarrow{t,\beta} (M_2, \alpha_2)$

$$Reach_{\mathcal{N}} = \{(M, \alpha) \mid (M_I, \alpha_I) \xrightarrow{*} (M, \alpha)\}$$

Abstraction technique - intuition

Intuitively, we build a new structure, called *constraint graph*, which abstracts multiple states of the reachability graph into a single state ("groups them together").

$$(\{p_0\}, \left\{\begin{matrix} \alpha(\mathtt{a}) = 0 \\ \alpha(\mathtt{b}) = 10 \end{matrix}\right\})$$

$$(\{p_1\}, \left\{\begin{matrix} \alpha(\mathtt{a}) = ... \\ \alpha(\mathtt{b}) = 10 \end{matrix}\right\})$$

$$(\{p_1\}, \left\{\begin{matrix} \alpha(\mathtt{a}) = ... \\ \alpha(\mathtt{b}) = 10 \end{matrix}\right\})$$

$$(\{p_1\}, \left\{\begin{matrix} \alpha(\mathtt{a}) = ... \\ \alpha(\mathtt{b}) = 10 \end{matrix}\right\})$$

$$(\{p_1\}, \left\{\begin{matrix} \alpha(\mathtt{a}) = ... \\ \alpha(\mathtt{b}) = 10 \end{matrix}\right\})$$

$\dots$

$\mathtt{a} > 5$

Abstraction technique - intuition

Our abstraction approach is not minimal, but it guarantees that, for each state that is “grouped together”:

- the set C of guards that are “accumulated” by firing transitions, up to that state, is satisfiable when seen as a constraint set;
- the marking in each state is the same;
- the same transitions are enabled.

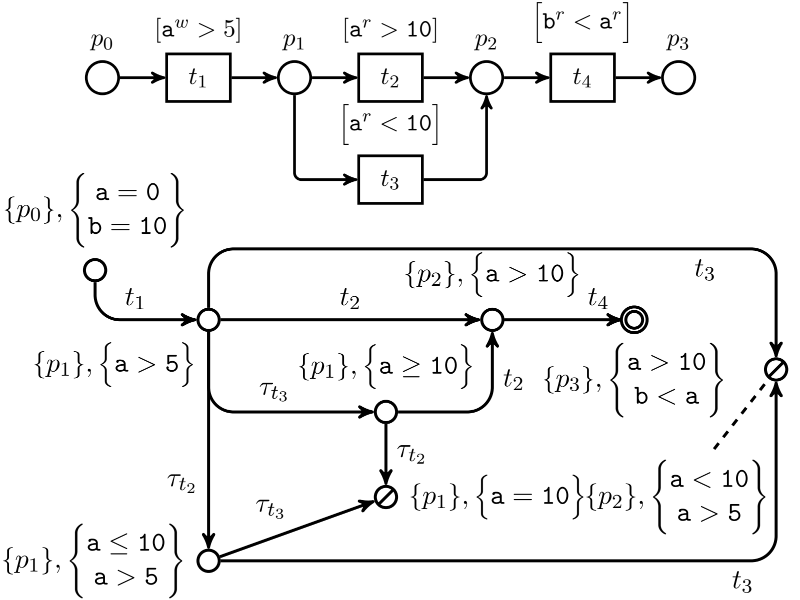
Constraint graph - lazy definition

Definition

The constraint graph $CG_{\mathcal{N}}$ of $\mathcal{N}$ is a tuple $\langle S, s_0, A\rangle$ where:

- $S \subseteq \mathcal{M} \times 2^{\mathcal{C}_V}$ is a set of states of the graph, which we call *nodes* to distinguish them from the notion of states of the DPN;
- $s_0 = (M_I, C_0) \in S$ is the initial node, where the initial constraints set is computed as $C_0 = \bigcup_{v \in V} \{v = \alpha_I(v)\}$;
- $A \subset S \times (T \cup \tau_T) \times S$ is the set of arcs, which is defined with S by mutual induction:
  - a transition $((M, C), t, (M', C'))$ is in A iff:
    - (i) $M[t\rangle M'$;
    - (ii) $C' = C \oplus guard(t)$ is satisfiable.
  - a transition $((M, C), \tau_t, (M, C''))$ is in A iff:
    - (i) $write(t) = \emptyset$;
    - (ii) $\exists M'$ s.t. $M[t\rangle M'$;
    - (iii) $C'' = C \oplus \neg guard(t)$ is satisfiable.

Example



p_0
$[\mathtt{a}^w > 5]$
t_1
p_1
$[\mathtt{a}^r > 10]$
t_2
p_2
$[\mathtt{b}^r < \mathtt{a}^r]$
t_4
p_3
$[\mathtt{a}^r < 10]$
t_3
$\{p_0\}, \left\{ \begin{array}{l} \mathtt{a} = 0 \\ \mathtt{b} = 10 \end{array} \right\}$
t_1
t_2
$\{p_2\}, \{\mathtt{a} > 10\}$
t_4
t_3
$\{p_1\}, \{\mathtt{a} > 5\}$
$\{p_1\}, \{\mathtt{a} \geq 10\}$
τ_{t_3}
t_2
$\{p_3\}, \left\{ \begin{array}{l} \mathtt{a} > 10 \\ \mathtt{b} < \mathtt{a} \end{array} \right\}$
τ_{t_2}
τ_{t_2}
τ_{t_3}
$\{p_1\}, \{\mathtt{a} = 10\}$
$\{p_2\}, \left\{ \begin{array}{l} \mathtt{a} < 10 \\ \mathtt{a} > 5 \end{array} \right\}$
$\{p_1\}, \left\{ \begin{array}{l} \mathtt{a} \leq 10 \\ \mathtt{a} > 5 \end{array} \right\}$
t_3

Constraint graph - lazy computation

$C_0 \leftarrow \bigcup_{v\in V} \{v = \alpha_I(v)\}$, $s_0 \leftarrow \langle M_I, C_0\rangle$, $S \leftarrow \{s_0\}$, $A \leftarrow \emptyset$, $L \leftarrow \{s_0\}$

while $L \neq \emptyset$ **do**

  $(M, C) \leftarrow \texttt{pick}(L)$

  $L \leftarrow L \setminus \{(M, C)\}$

  foreach $t \in T$ *s.t.* $M \xrightarrow{t} M'$ **do**

    $C' \leftarrow C \oplus guard(t)$

    $C'' \leftarrow C$

    if $write(t) = \emptyset$ **then**

      $C'' \leftarrow C'' \oplus \neg guard(t)$

    if $\texttt{satisfiable}(C')$ **then**

      if $\exists(\bar{M}, \bar{C}) \in S$ *s.t.* $M' > \bar{M} \wedge C' = \bar{C}$ **then** //The net is unbounded

        return false

      $S \leftarrow S \cup \{(M', C')\}$

      $A \leftarrow A \cup \{\langle (M, C), t, (M', C')\rangle\}$

      $L \leftarrow L \cup \{(M', C')\}$

    if $\texttt{satisfiable}(C'') \wedge C \neq C''$ **then**

      $S \leftarrow S \cup \{(M, C'')\}$

      $A \leftarrow A \cup \{\langle (M, C), \tau_t, (M, C'')\rangle\}$

      $L \leftarrow L \cup \{(M, C'')\}$

return $\texttt{analyzeConstraintGraph}\ (\langle S, s_0, A\rangle)$

Main result

Theorem

$RG_{\mathcal{N}}$ is data-aware sound iff $CG_{\mathcal{N}}$ is data-aware sound.

The obtained structure is not bisimilar to the original DPN $\mathcal{N}$, but is data-aware sound iff $\mathcal{N}$ is so. Crucially, the new state space is **finite**.

Main result

Theorem

$RG_{\mathcal{N}}$ is data-aware sound iff $CG_{\mathcal{N}}$ is data-aware sound.

The obtained structure is not bisimilar to the original DPN $\mathcal{N}$, but is data-aware sound iff $\mathcal{N}$ is so. Crucially, the new state space is **finite**.

So…

- Decidability by reduction to finite-state reachability graph analysis.
- Practical and implementable procedure for doing so.
- Already implemented for DPNs that only use variable-to-constant guards.

Main result

Theorem

$RG_{\mathcal{N}}$ is data-aware sound iff $CG_{\mathcal{N}}$ is data-aware sound.

The obtained structure is not bisimilar to the original DPN $\mathcal{N}$, but is data-aware sound iff $\mathcal{N}$ is so. Crucially, the new state space is **finite**.

So...

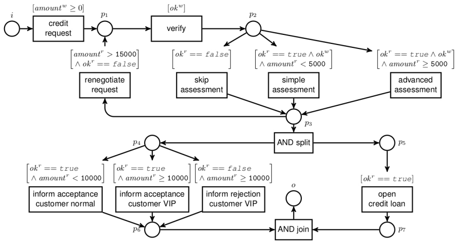
- Decidability by reduction to finite-state reachability graph analysis.
- Practical and implementable procedure for doing so.
- Already implemented for DPNs that only use variable-to-constant guards.

Generality of the result

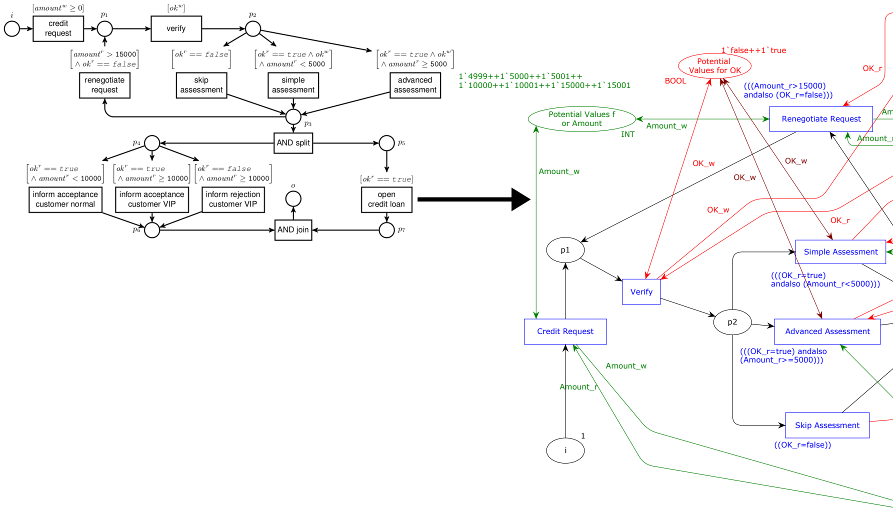
Our technique extends to any constraint language that:

- generates only boundedly many constraints over a fixed set of variables and constants, and
- has decidable satisfiability.

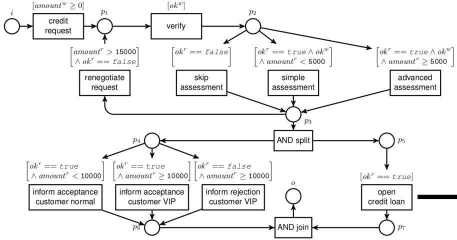
Analysis of DPNs with variable-to-constant guards

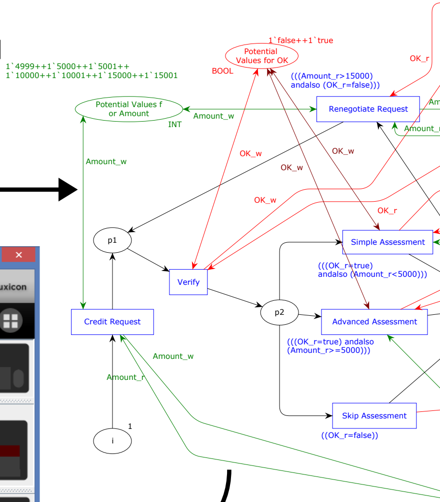


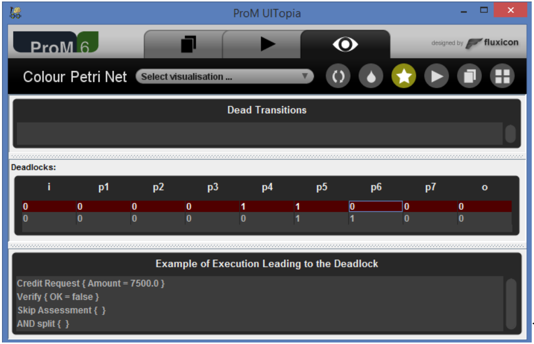
Analysis of DPNs with variable-to-constant guards



Analysis of DPNs with variable-to-constant guards





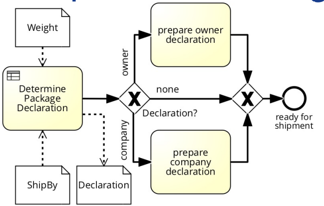


Back to our setting. . .

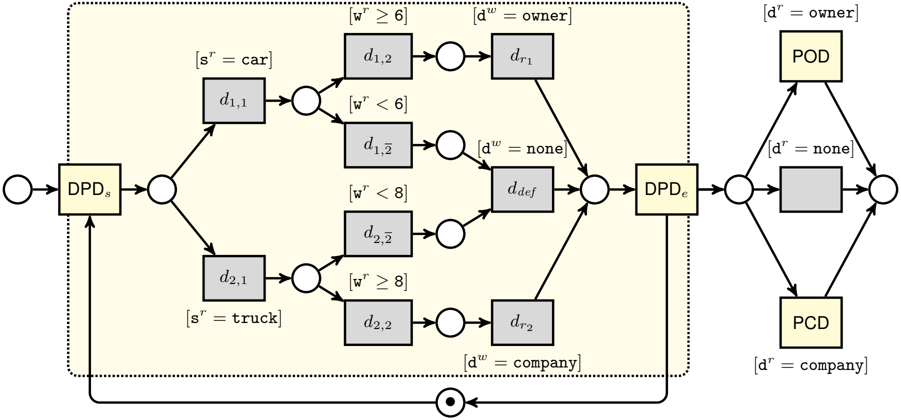
From BPMN with Case Data and S-FEEL decisions to DPN

1. Pre-processing of decision tables:
   a. Uniqueification [Batoulis and Weske, BPMDemo2018].
   b. Completion: adding complementary rules to handle default values (or special output `undefined`).
2. control-flow $\rightarrow$ P/T net. [Standard techniques]
3. Data objects $\rightarrow$ variables.
4. I/O connectors $\rightarrow$ read-write guards.
5. Decisions and service tasks $\rightarrow$ non-interruptible circuit sub-net with read-write guards.

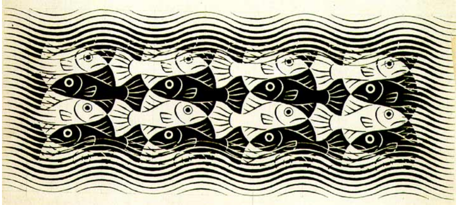
Example of encoding



Package Declaration			
U	ShipBy	Weight (kg)	Declaration
	car,truck	> 0	none, owner, company
1	car	≥ 6	owner
2	truck	≥ 8	company



Conclusions



Diversify

Importance of multi-perspective models with solid foundations.

Contextualize

Background knowledge and processes to put decisions in perspective.

Cross-fertilize

Solid formal foundations and effective analysis techniques by mixing:

conceptual modeling   formal methods   artificial intelligence (KR)

Future work

Strategic reasoning with multiple decision-makers

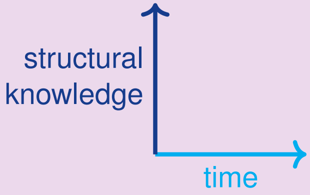
We are extending our abstraction technique to:

- verify arbitrary linear temporal properties of DPNs;
- automatically compute a witness for these properties;
- also in the presence of different actors controlling choice points and variable assignments.

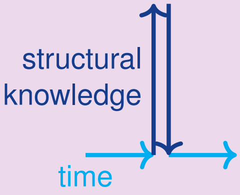
Combining processes, decisions, and background knowledge

Two different settings, depending on how time and knowledge interact:

Two-dimensional reasoning



Levesque functional approach



Thanks for listening!

A big thanks to

Diego Calvanese Massimiliano de Leoni
Marlon Dumas Paolo Felli
Fabrizio Maggi