From DB-nets to Coloured Petri Nets with Priorities

Marco Montali and **Andrey Rivkin**

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy

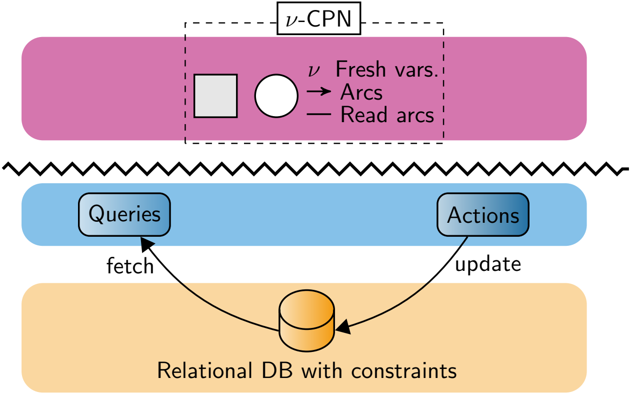
Process-data dichotomy

- A well-known problem coming from the BPM community

- The leitmotiv: *how to make processes and data work together?*

Process-data dichotomy

- **Two research streams** that address the dichotomy
  - Petri nets: enrich PNs with some form of data that accounts for, e.g., fresh ID of objects
  - Databases: enrich DBs with actions



ν-coloured Petri nets

- Almost like standard CPNs

ν-coloured Petri nets

- Almost like standard CPNs
- *Colours* $\rightsquigarrow$ data types $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \Gamma_{\mathcal{D}} \rangle$ from $\mathfrak{D}$ (a finite set of types)
  - $\Delta_{\mathcal{D}}$ – a value domain (could be infinite!)
  - $\Gamma_{\mathcal{D}}$ – a finite set of *predicate symbols*
  - Examples: **string** $= \langle \mathbb{S}, \{=_s\} \rangle$, **int** $= \langle \mathbb{Z}, \{=_{int}, <_{int}, succ\} \rangle$

ν-coloured Petri nets

- Almost like standard CPNs
- *Colours* $\rightsquigarrow$ data types $\mathcal{D} = \langle \Delta_{\mathcal{D}}, \Gamma_{\mathcal{D}} \rangle$ from $\mathfrak{D}$ (a finite set of types)
  - $\Delta_{\mathcal{D}}$ – a value domain (could be infinite!)
  - $\Gamma_{\mathcal{D}}$ – a finite set of *predicate symbols*
  - Examples: **string** $= \langle \mathbb{S}, \{=_s\} \rangle$, **int** $= \langle \mathbb{Z}, \{=_{int}, <_{int}, succ\} \rangle$
- *Arc inscriptions* have no complex expressions, only variables

ν-coloured Petri nets

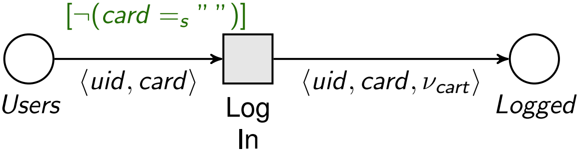
- Almost like standard CPNs
- *Colours* $\rightsquigarrow$ data types $\mathcal{D} = \langle \Delta_\mathcal{D}, \Gamma_\mathcal{D} \rangle$ from $\mathfrak{D}$ (a finite set of types)
    - $\Delta_\mathcal{D}$ – a value domain (could be infinite!)
    - $\Gamma_\mathcal{D}$ – a finite set of *predicate symbols*
    - Examples: **string** $= \langle \mathbb{S}, \{=_s\} \rangle$, **int** $= \langle \mathbb{Z}, \{=_{int}, <_{int}, succ\} \rangle$
- *Arc inscriptions* have no complex expressions, only variables
- Two kinds of *(typed) variables*:
    - $\mathcal{V}_\mathfrak{D}$ – "normal" variables
    - $\Upsilon_\mathfrak{D}$ – fresh variables (a la ν-PNs)
    - **unbounded variables** in the output arc expressions account for external input and fresh data

ν-coloured Petri nets

- Almost like standard CPNs
- *Colours* $\rightsquigarrow$ data types $\mathcal{D} = \langle \Delta_\mathcal{D}, \Gamma_\mathcal{D} \rangle$ from $\mathfrak{D}$ (a finite set of types)
  - $\Delta_\mathcal{D}$ – a value domain (could be infinite!)
  - $\Gamma_\mathcal{D}$ – a finite set of *predicate symbols*
  - Examples: **string** $= \langle \mathbb{S}, \{=_s\} \rangle$, **int** $= \langle \mathbb{Z}, \{=_{int}, <_{int}, succ\} \rangle$
- *Arc inscriptions* have no complex expressions, only variables
- Two kinds of *(typed) variables*:
  - $\mathcal{V}_\mathfrak{D}$ – "normal" variables
  - $\Upsilon_\mathfrak{D}$ – fresh variables (a la ν-PNs)
  - **unbounded variables** in the output arc expressions account for external input and fresh data
- *Guards*: quantifier- and relation-free FO formulas over $\mathcal{D}$'s

ν-coloured Petri nets

A simple net for logging in users in an online shop



$[\neg(\mathit{card} =_s "\,")]$

Users

$\langle \mathit{uid}, \mathit{card} \rangle$

Log In

$\langle \mathit{uid}, \mathit{card}, \nu_{\mathit{cart}} \rangle$

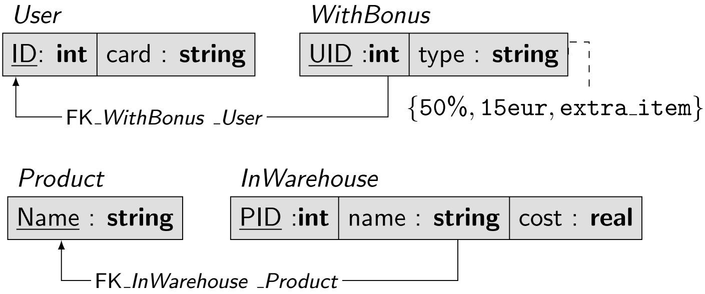
Logged

$$\begin{aligned} \texttt{color}(\mathit{Users}) &= \textbf{int} \times \textbf{string} \\ \texttt{color}(\mathit{Logged}) &= \textbf{int} \times \textbf{string} \times \textbf{int} \end{aligned}$$

- *"... log in only if you have credit card data"*
- $\nu_{\mathit{cart}} \in \Upsilon_{\mathfrak{D}}$ is used to create a (globally) new shopping cart ID

Relational Database: schema

A simplified online shop database



User

ID: **int** | card : **string**

WithBonus

UID :**int** | type : **string**

$\{\texttt{50\%}, \texttt{15eur}, \texttt{extra\_item}\}$

FK_*WithBonus* _*User*

Product

Name : **string**

InWarehouse

PID :**int** | name : **string** | cost : **real**

FK_*InWarehouse* _*Product*

- A user may have only(!) one out of three predefined bonuses
- *Product* stores types of products available in the online shop

Relational Database: queries

- **Queries** – FO expressions over $\mathfrak{D}$-typed DB schema $\mathcal{R}$ with explicitly identified free (*answer*) variables
- Examples:
  - *"get all products available in the warehouse and whose price has been defined"*

    $$\mathtt{Q}_{\mathtt{products}}(pid, n, c)\text{:- } Product(n) \wedge InWarehouse(pid, n, c) \wedge c \neq \mathtt{null}$$

  - *"get all registered users"*

    $$\mathtt{Q}_{\mathtt{users}}(uid)\text{:- } \exists card.User(id, card)$$

  - *"get all bonus holders"*

    $$\mathtt{Q}_{\mathtt{wbonus}}(uid, bt')\text{:- } WithBonus(uid, bt')$$

Relational Database: updates

- . . . via **parametrized atomic actions**

Relational Database: updates

- . . . via **parametrized atomic actions**
- Specify 1st which facts to delete and 2nd which facts to add
  - Like in STRIPS planning
  - Follow the order ⇒ avoid situations in which one fact is both added and deleted
- Actions are transactional
  - If an action application result violates database constraints ⇒ rollback!

Relational Database: updates

How to **assign** *a bonus to a user?* Use an action $\textsc{AddB}(uid, bt)$ s.t.

- $\textsc{AddB}{\cdot}\texttt{del} = \emptyset$
- $\textsc{AddB}{\cdot}\texttt{add} = \{WithBonus(uid, bt)\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
–	–

Relational Database: updates

How to **assign** *a bonus to a user?* Use an action $\textsc{AddB}(\mathit{uid}, \mathit{bt})$ s.t.

- $\textsc{AddB}\cdot\texttt{del} = \emptyset$
- $\textsc{AddB}\cdot\texttt{add} = \{\mathit{WithBonus}(\mathit{uid}, \mathit{bt})\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

1. execute $\textsc{AddB}(122, 50\%)$

WithBonus

UID	type
–	–

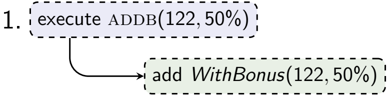
Relational Database: updates

How to **assign** *a bonus to a user?* Use an action $\textsc{addb}(\mathit{uid}, \mathit{bt})$ s.t.

- $\textsc{addb}{\cdot}\mathtt{del} = \emptyset$
- $\textsc{addb}{\cdot}\mathtt{add} = \{\mathit{WithBonus}(\mathit{uid}, \mathit{bt})\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923



1. execute $\textsc{addb}(122, 50\%)$

add $\mathit{WithBonus}(122, 50\%)$

WithBonus

UID	type
122	50%

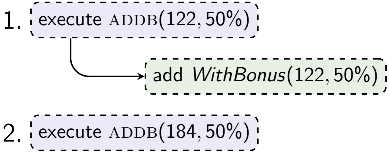
Relational Database: updates

How to **assign** *a bonus to a user?* Use an action $\textsc{addb}(uid, bt)$ s.t.

- $\textsc{addb}\cdot\mathtt{del} = \emptyset$
- $\textsc{addb}\cdot\mathtt{add} = \{WithBonus(uid, bt)\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923



1. execute $\textsc{addb}(122, 50\%)$

add $WithBonus(122, 50\%)$

2. execute $\textsc{addb}(184, 50\%)$

WithBonus

UID	type
122	50%

Relational Database: updates

How to **assign** *a bonus to a user?* Use an action $\textsc{addb}(\mathit{uid}, \mathit{bt})$ s.t.

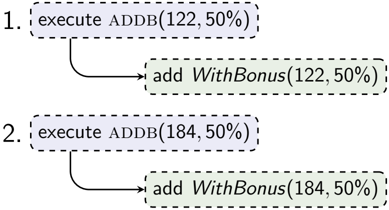
- $\textsc{addb}\cdot\mathtt{del} = \emptyset$
- $\textsc{addb}\cdot\mathtt{add} = \{\mathit{WithBonus}(\mathit{uid}, \mathit{bt})\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%
184	50%



1. execute $\textsc{addb}(122, 50\%)$

add $\mathit{WithBonus}(122, 50\%)$

2. execute $\textsc{addb}(184, 50\%)$

add $\mathit{WithBonus}(184, 50\%)$

Relational Database: updates

How to **assign** *a bonus to a user?* Use an action $\textsc{AddB}(uid, bt)$ s.t.

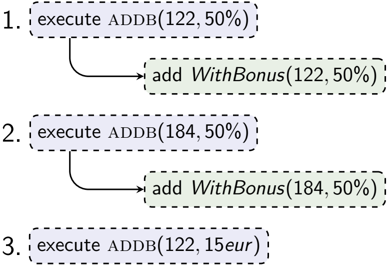
- $\textsc{AddB}{\cdot}\mathtt{del} = \emptyset$
- $\textsc{AddB}{\cdot}\mathtt{add} = \{\mathit{WithBonus}(uid, bt)\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%
184	50%



1. execute $\textsc{AddB}(122, 50\%)$ → add $\mathit{WithBonus}(122, 50\%)$
2. execute $\textsc{AddB}(184, 50\%)$ → add $\mathit{WithBonus}(184, 50\%)$
3. execute $\textsc{AddB}(122, 15\mathit{eur})$

Relational Database: updates

How to **assign** *a bonus to a user?* Use an action $\textsc{addb}(uid, bt)$ s.t.

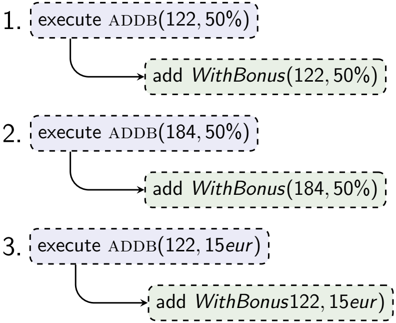
- $\textsc{addb}{\cdot}\texttt{del} = \emptyset$
- $\textsc{addb}{\cdot}\texttt{add} = \{WithBonus(uid, bt)\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%
184	50%
122	15eur



1. execute $\textsc{addb}(122, 50\%)$

   add $WithBonus(122, 50\%)$

2. execute $\textsc{addb}(184, 50\%)$

   add $WithBonus(184, 50\%)$

3. execute $\textsc{addb}(122, 15eur)$

   add $WithBonus122, 15eur)$

Relational Database: updates

How to **assign** *a bonus to a user?* Use an action $\textsc{addb}(\mathit{uid}, \mathit{bt})$ s.t.

- $\textsc{addb}\cdot\texttt{del} = \emptyset$
- $\textsc{addb}\cdot\texttt{add} = \{\mathit{WithBonus}(\mathit{uid}, \mathit{bt})\}$

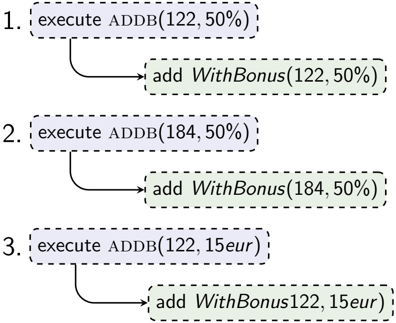
User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%
184	50%
122	15eur

constraint vioaltion:
"only one bonus per user"



1. execute $\textsc{addb}(122, 50\%)$
   add $\mathit{WithBonus}(122, 50\%)$
2. execute $\textsc{addb}(184, 50\%)$
   add $\mathit{WithBonus}(184, 50\%)$
3. execute $\textsc{addb}(122, 15\mathit{eur})$
   add $\mathit{WithBonus}122, 15\mathit{eur})$

Relational Database: updates

How to **assign** *a bonus to a user?* Use an action $\textsc{AddB}(uid, bt)$ s.t.

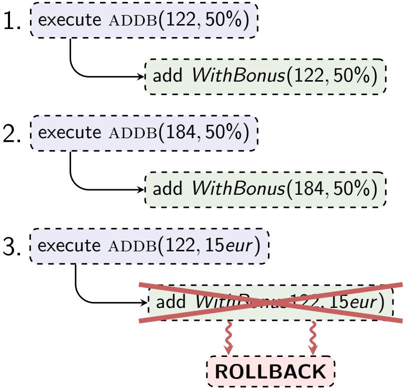
- $\textsc{AddB}{\cdot}\mathtt{del} = \emptyset$
- $\textsc{AddB}{\cdot}\mathtt{add} = \{\mathit{WithBonus}(uid, bt)\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%
184	50%
~~122~~	~~15eur~~



1. execute $\textsc{AddB}(122, 50\%)$
   add $\mathit{WithBonus}(122, 50\%)$
2. execute $\textsc{AddB}(184, 50\%)$
   add $\mathit{WithBonus}(184, 50\%)$
3. execute $\textsc{AddB}(122, 15\mathit{eur})$
   ~~add $\mathit{WithBonus}(122, 15\mathit{eur})$~~

ROLLBACK

Relational Database: updates

How to **assign** *a bonus to a user?* Use an action $\textsc{addb}(\mathit{uid}, \mathit{bt})$ s.t.

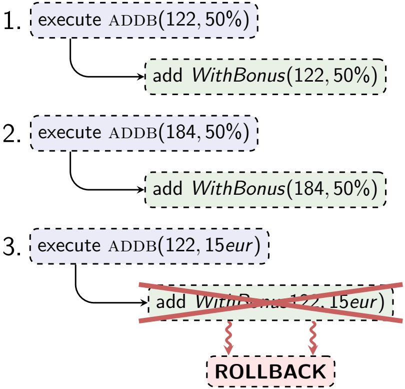
- $\textsc{addb}{\cdot}\mathtt{del} = \emptyset$
- $\textsc{addb}{\cdot}\mathtt{add} = \{\mathit{WithBonus}(\mathit{uid}, \mathit{bt})\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%
184	50%



1. execute $\textsc{addb}(122, 50\%)$

add $\mathit{WithBonus}(122, 50\%)$

2. execute $\textsc{addb}(184, 50\%)$

add $\mathit{WithBonus}(184, 50\%)$

3. execute $\textsc{addb}(122, 15\mathit{eur})$

~~add $\mathit{WithBonus}(122, 15\mathit{eur})$~~

ROLLBACK

Relational Database: updates

How to **change** *a user's bonus?* Use an action $\textsc{change}(uid, bt, bt')$ s.t.

- $\textsc{change}\cdot\texttt{del} = \{WithBonus(uid, bt')\}$
- $\textsc{change}\cdot\texttt{add} = \{WithBonus(uid, bt)\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%
184	50%

Relational Database: updates

How to **change** *a user's bonus?* Use an action $\textsc{change}(uid, bt, bt')$ s.t.

- $\textsc{change}{\cdot}\mathtt{del} = \{\mathit{WithBonus}(uid, bt')\}$
- $\textsc{change}{\cdot}\mathtt{add} = \{\mathit{WithBonus}(uid, bt)\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

execute $\textsc{change}(122, 15eur, 50\%)$

WithBonus

UID	type
122	50%
184	50%

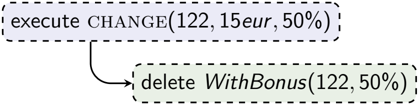
Relational Database: updates

How to **change** *a user's bonus?* Use an action $\textsc{Change}(uid, bt, bt')$ s.t.

- $\textsc{Change}\cdot\mathtt{del} = \{WithBonus(uid, bt')\}$
- $\textsc{Change}\cdot\mathtt{add} = \{WithBonus(uid, bt)\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923



execute $\textsc{Change}(122, 15eur, 50\%)$

delete $WithBonus(122, 50\%)$

WithBonus

UID	type
184	50%

Relational Database: updates

How to **change** *a user's bonus?* Use an action $\textsc{change}(\mathit{uid}, \mathit{bt}, \mathit{bt}')$ s.t.

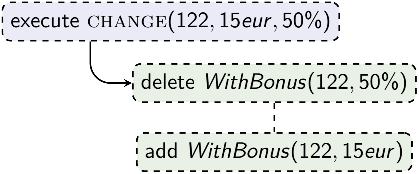
- $\textsc{change}{\cdot}\mathtt{del} = \{\mathit{WithBonus}(\mathit{uid}, \mathit{bt}')\}$
- $\textsc{change}{\cdot}\mathtt{add} = \{\mathit{WithBonus}(\mathit{uid}, \mathit{bt})\}$

User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	15eur
184	50%



execute $\textsc{change}(122, 15\mathit{eur}, 50\%)$

delete $\mathit{WithBonus}(122, 50\%)$

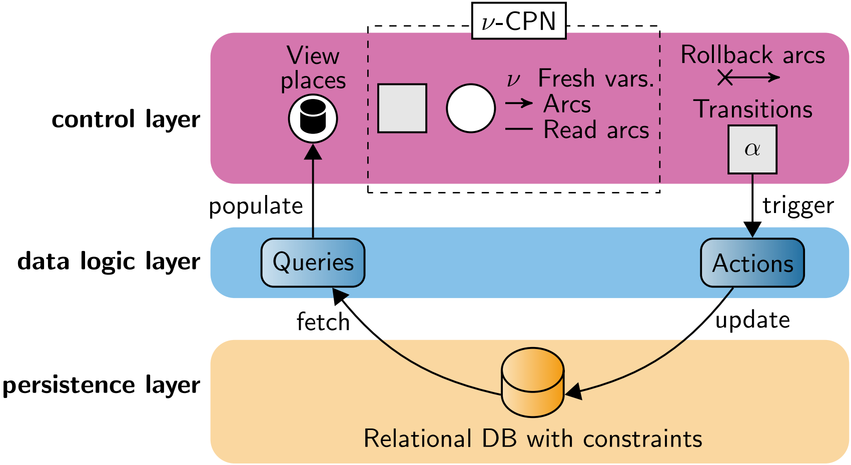
add $\mathit{WithBonus}(122, 15\mathit{eur})$

DB-nets

How to account for ν-CPNs and DBs $+$ jointly respect semantics of both?

DB-nets

How to account for ν-CPNs and DBs + jointly respect semantics of both?

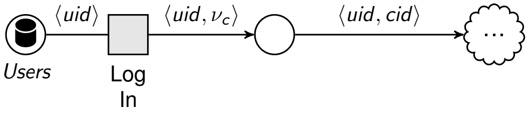


A missing bit: view places

- "Views" over the persistence layer
- **Host answers to queries** from the data logic
- Clearly identify where the control layer **"reads"** from the **persistence layer**
- Not possible to **explicitly modify** by the control layer...
- ...but can be **implicitly modified** by applying actions on the persistence layer and recomputing the view

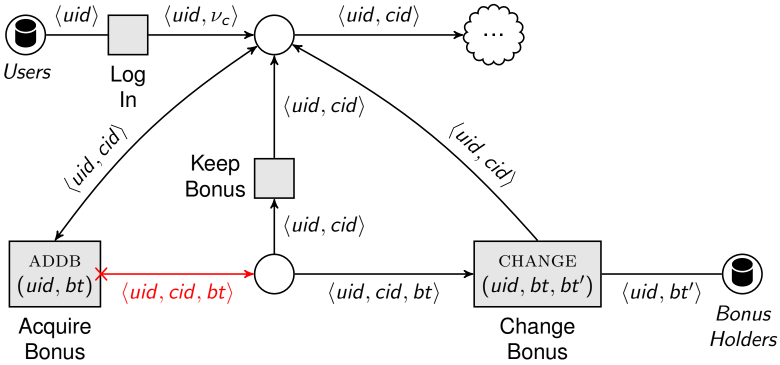
A (partial) DB-net example

- Part #1: a simple net for logging in users in an online shop
  - A view place *Users* is equipped with query $\mathtt{Q}_{\mathtt{users}}$

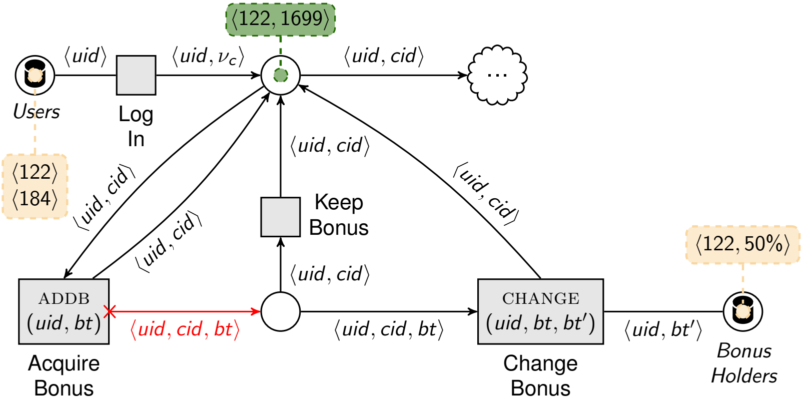


A (partial) DB-net example

- Part#2: a net for managing user bonuses
  - AcquireBonus and ChangeBonus have actions assigned to them


$\langle uid \rangle$
$\langle uid, \nu_c \rangle$
$\langle uid, cid \rangle$
...
Users
Log In
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
Keep Bonus
$\langle uid, cid \rangle$
ADDB (uid, bt)
$\langle uid, cid, bt \rangle$
$\langle uid, cid, bt \rangle$
CHANGE (uid, bt, bt')
$\langle uid, bt' \rangle$
Acquire Bonus
Change Bonus
Bonus Holders

Firing of transitions


$\langle 122, 1699 \rangle$
$\langle uid \rangle$
$\langle uid, \nu_c \rangle$
$\langle uid, cid \rangle$
...
Users
Log In
$\langle 122 \rangle$
$\langle 184 \rangle$
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
Keep Bonus
$\langle uid, cid \rangle$
$\langle 122, 50\% \rangle$
ADDB
(uid, bt)
$\langle uid, cid, bt \rangle$
$\langle uid, cid, bt \rangle$
CHANGE
(uid, bt, bt')
$\langle uid, bt' \rangle$
Bonus Holders
Acquire Bonus
Change Bonus

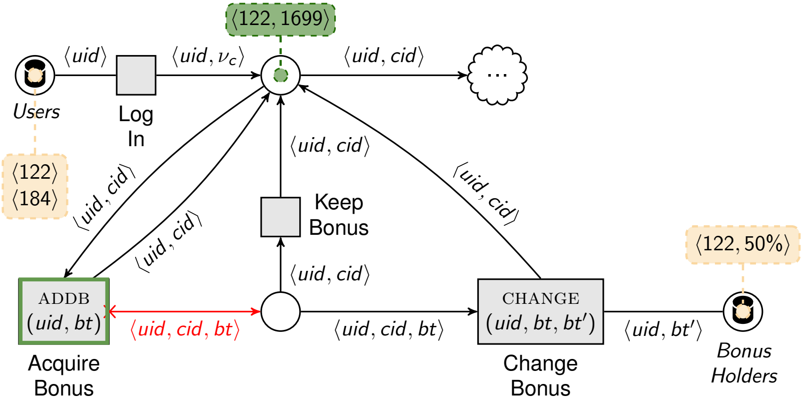
User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%

Firing of transitions



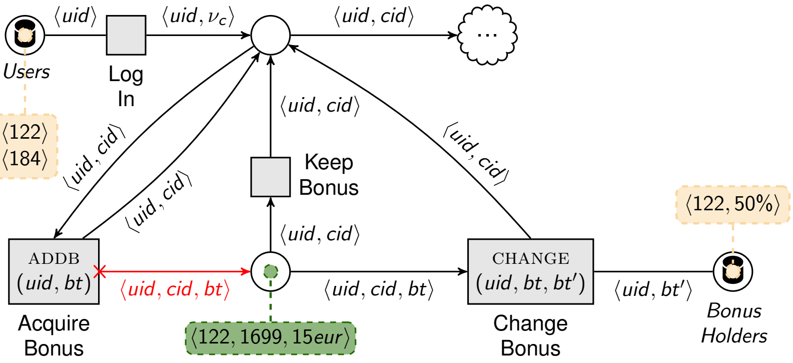
User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%

Firing of transitions


$\langle uid \rangle$
$\langle uid, \nu_c \rangle$
$\langle uid, cid \rangle$
…
Users
Log In
$\langle 122 \rangle$
$\langle 184 \rangle$
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
Keep Bonus
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
$\langle 122, 50\% \rangle$
ADDB (uid, bt)
$\langle uid, cid, bt \rangle$
$\langle uid, cid, bt \rangle$
CHANGE (uid, bt, bt')
$\langle uid, bt' \rangle$
Acquire Bonus
$\langle 122, 1699, 15eur \rangle$
Change Bonus
Bonus Holders

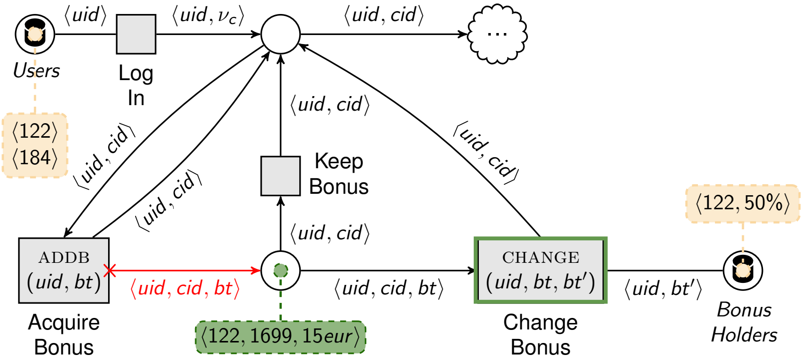
User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%

Firing of transitions


$\langle uid \rangle$
Log In
$\langle uid, \nu_c \rangle$
$\langle uid, cid \rangle$
$\cdots$
Users
$\langle 122 \rangle$
$\langle 184 \rangle$
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
Keep Bonus
$\langle uid, cid \rangle$
$\langle uid, cid \rangle$
$\langle 122, 50\% \rangle$
ADDB (uid, bt)
Acquire Bonus
$\langle uid, cid, bt \rangle$
$\langle 122, 1699, 15eur \rangle$
$\langle uid, cid, bt \rangle$
CHANGE (uid, bt, bt')
Change Bonus
$\langle uid, bt' \rangle$
Bonus Holders

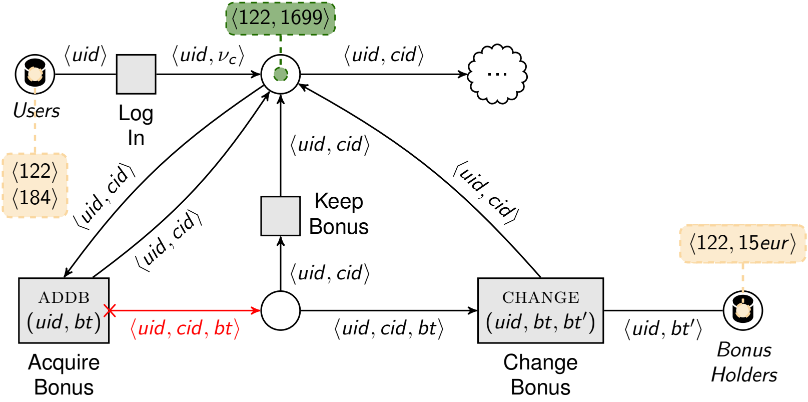
User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	50%

Firing of transitions



User

ID	card
122	5583-3290-2131-2333
184	4419-2311-1189-9923

WithBonus

UID	type
122	15eur

DB-nets

- We know how to model and simulate DB-nets using CPN Tools + Access/CPN + Comms/CPN...
  - External libraries allow to fully account for actions, view places and DB interactions
  - We also know how to tame the infinity achieving decidabiluty of verification in relevant fragments

DB-nets

- We know how to model and simulate DB-nets using CPN Tools + Access/CPN + Comms/CPN...
- ...but we cannot correctly generate state spaces due to limitations of Access/CPN
  - The content of view places is changed by actions and not properly recomputed after each transition firing

DB-nets

- We know how to model and simulate DB-nets using CPN Tools + Access/CPN + Comms/CPN...
- ...but we cannot correctly generate state spaces due to limitations of Access/CPN
    - The content of view places is changed by actions and not properly recomputed after each transition firing

Is it possible to avoid view places (and even actions)?

From DB-nets to ν-CPNs

- We can fully "lift" DB-nets to CPN Tools
- That is, we **map the entire DB and its management into CPN Tools**

Any limitations on queries and relational DB + constraints?

From DB-nets to ν-CPNs

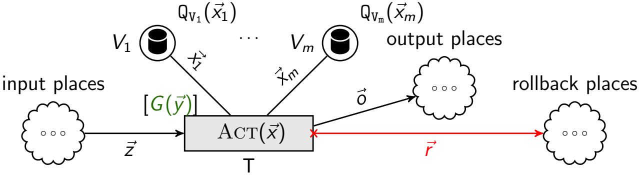
- We can fully "lift" DB-nets to CPN Tools
- That is, we **map the entire DB and its management into CPN Tools**

Any limitations on queries and relational DB + constraints?

- Stay on the safe side: DB-nets with UCQs$^{\neq}$, DB with PK, FK and CHECK
  - UCQs$^{\neq}$ correspond to SELECT-FROM-WHERE SQL queries
  - PKs, FKs and domain constraints are just easy to manage ☺

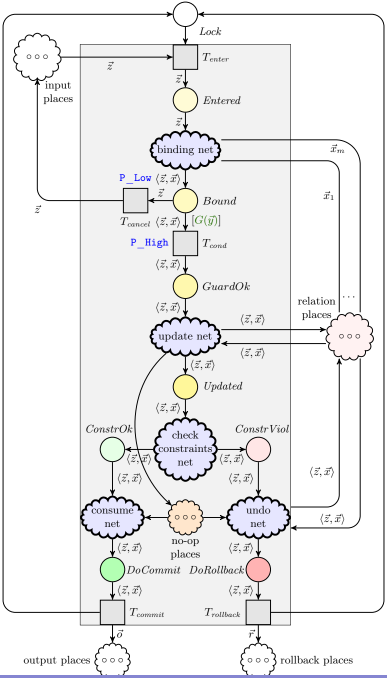
Result: a **translation** into ν-CPNs with priorities and extensive support of SML (supported by CPN Tools)

Translation



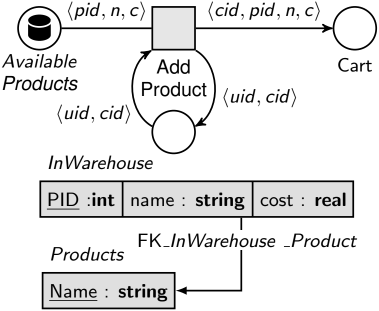
- A database is represented using *relational places*
- Other DB-net elements are **actually computed** on transition T firing in 4 phases:
    1. collect variable bindings and compute the content of view places adjacent to T
    2. if there is an action assigned to T, execute it
    3. check the satisfaction of integrity constraints
    4. finish the computation and generate a new marking
- To realize the execution of original T, all the four phases are executed uninterruptedly (under global lock)

Translation



Computing views using ν-CPN places

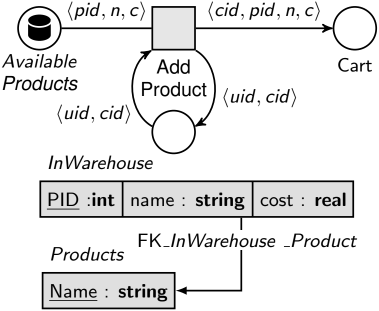
An original DB-net



$\mathtt{Q}_{\mathtt{products}}(pid, n, c)):-$
$Product(n) \wedge InWarehouse(pid, n, c) \wedge$
$\wedge c \neq \mathtt{null}$

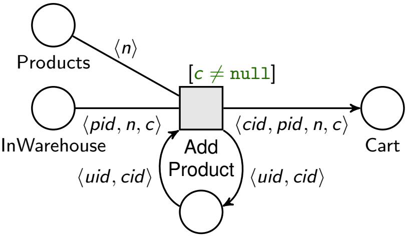
Computing views using ν-CPN places

An original DB-net



$\mathtt{Q_{products}}(pid, n, c)):\!-$
$Product(n) \wedge InWarehouse(pid, n, c)\wedge$
$\wedge c \neq \mathtt{null}$

An intuitive ν-CPN encoding

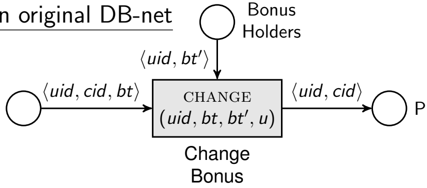


Products
$\langle n\rangle$
$[c \neq \mathtt{null}]$
InWarehouse
$\langle pid, n, c\rangle$
$\langle cid, pid, n, c\rangle$
Cart
Add Product
$\langle uid, cid\rangle$
$\langle uid, cid\rangle$

- Products and InWarehouse are relational places
- $\mathtt{Q_{products}}$ is represented using relational places + a guard

Modelling RDBMS updates in ν-CPNs

An original DB-net

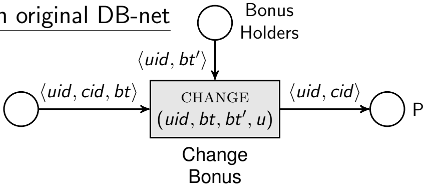

Bonus
Holders
$\langle uid, bt' \rangle$
$\langle uid, cid, bt \rangle$
CHANGE
(uid, bt, bt', u)
$\langle uid, cid \rangle$
P
Change
Bonus

CHANGE(uid, bt, bt'):

- CHANGE$\cdot$`del` $= \{WithBonus(uid, bt')\}$
- CHANGE$\cdot$`add` $= \{WithBonus(uid, bt)\}$

Modelling RDBMS updates in ν-CPNs

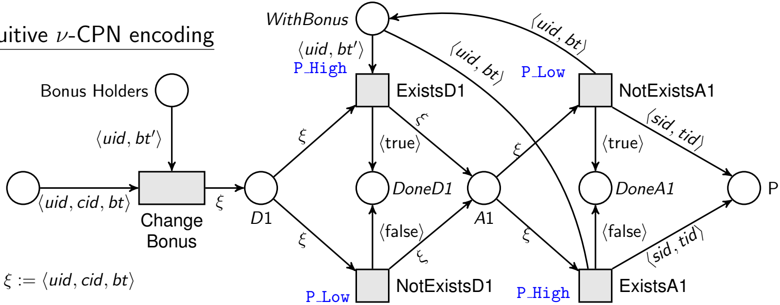
An original DB-net


Bonus
Holders
$\langle uid, bt' \rangle$
$\langle uid, cid, bt \rangle$
CHANGE
(uid, bt, bt', u)
$\langle uid, cid \rangle$
P
Change
Bonus

$\textsc{change}(uid, bt, bt')$:

- $\textsc{change}\cdot\mathtt{del} = \{WithBonus(uid, bt')\}$
- $\textsc{change}\cdot\mathtt{add} = \{WithBonus(uid, bt)\}$

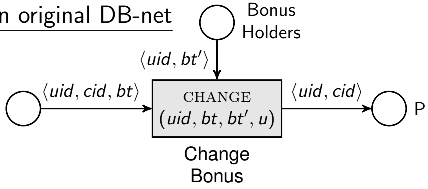
An intuitive ν-CPN encoding



Preserve update (and set) semantics via prioritized transitions that check if a tuple to add/delete already exists in a relation place

Modelling RDBMS updates in ν-CPNs

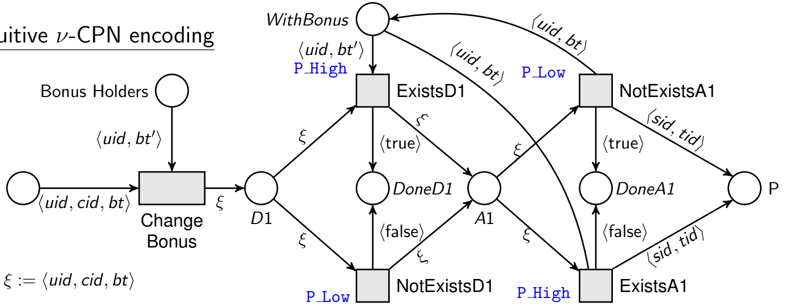
An original DB-net


Bonus Holders
$\langle uid, bt' \rangle$
$\langle uid, cid, bt \rangle$
CHANGE (uid, bt, bt', u)
$\langle uid, cid \rangle$
P
Change Bonus

$\textsc{change}(uid, bt, bt')$:

- $\textsc{change}\cdot\texttt{del} = \{WithBonus(uid, bt')\}$
- $\textsc{change}\cdot\texttt{add} = \{WithBonus(uid, bt)\}$

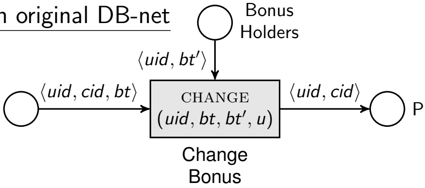
An intuitive ν-CPN encoding



Auxiliary *Done*-places: if **true**, then the token has been successfully added/deleted; **false** otherwise

Modelling RDBMS updates in ν-CPNs

An original DB-net

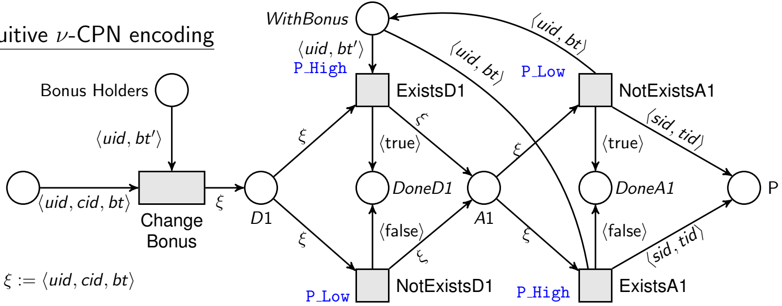


Bonus Holders
$\langle uid, bt' \rangle$
$\langle uid, cid, bt \rangle$
CHANGE (uid, bt, bt', u)
$\langle uid, cid \rangle$
P
Change Bonus

CHANGE(uid, bt, bt'):

- CHANGE$\cdot$`del` $= \{WithBonus(uid, bt')\}$
- CHANGE$\cdot$`add` $= \{WithBonus(uid, bt)\}$

An intuitive ν-CPN encoding

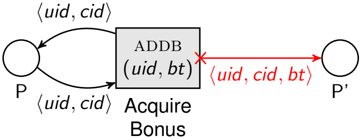


$\xi := \langle uid, cid, bt \rangle$

The **update execution order** is **the same** as for DB-nets

Checking integrity constraints and getting a new marking

An original DB-net

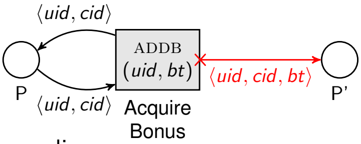

$\langle uid, cid\rangle$
P
$\langle uid, cid\rangle$
ADDB
(uid, bt)
Acquire Bonus
$\langle uid, cid, bt\rangle$
P'

$\textsc{AddB}(uid, bt)$:

- $\textsc{AddB}\cdot\texttt{del} = \emptyset$
- $\textsc{AddB}\cdot\texttt{add} = \{WithBonus(uid, bt)\}$

Checking integrity constraints and getting a new marking

An original DB-net

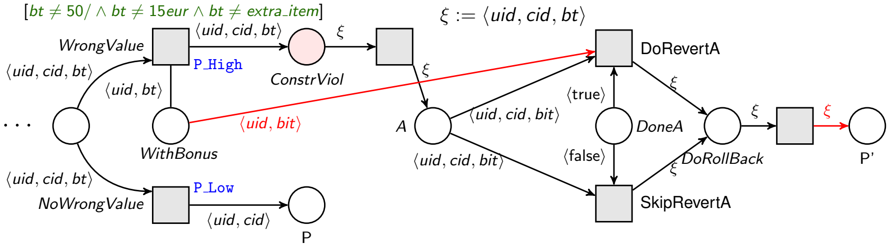


$\langle uid, cid\rangle$
ADDB
(uid, bt)
P
$\langle uid, cid\rangle$
Acquire Bonus
$\langle uid, cid, bt\rangle$
P'

ADDB(uid, bt):

- ADDB$\cdot$**del** $= \emptyset$
- ADDB$\cdot$**add** $= \{WithBonus(uid, bt)\}$

An intuitive ν-CPN encoding



$[bt \neq 50/ \wedge bt \neq 15eur \wedge bt \neq extra\_item]$
$\xi := \langle uid, cid, bt\rangle$
WrongValue
$\langle uid, cid, bt\rangle$
P_High
ξ
ConstrViol
$\langle uid, cid, bt\rangle$
$\langle uid, bt\rangle$
ξ
DoRevertA
$\langle true\rangle$
ξ
. . .
WithBonus
$\langle uid, bit\rangle$
A
$\langle uid, cid, bit\rangle$
DoneA
ξ
ξ
$\langle uid, cid, bt\rangle$
$\langle uid, cid, bit\rangle$
$\langle false\rangle$
DoRollBack
ξ
P'
NoWrongValue
P_Low
$\langle uid, cid\rangle$
P
SkipRevertA

- Check integrity constraints
- If any *violated*, **rollback** all the **successfully performed** (i.e., marked with true in *Done*-places) **updates**

Results

- A fragment of DB-nets with unions of conjunctive queries with negative filters can be translated into ν-CPNs with transition priorities
- The translation produces a net that is bisimilar to the original one

Results

- A fragment of DB-nets with unions of conjunctive queries with negative filters can be translated into ν-CPNs with transition priorities
- The translation produces a net that is bisimilar to the original one
- What to do with this result?
  - Modelling and analyzing data-intensive applications in CPN Tools
  - Study concurrency in databases
  - Implement the translation in the DB-net extension
  - Add more support for different types of integrity constraints
  - Make our state-space abstraction technique operational in CPN Tools

Questions, please

ARE THERE ANY QUESTIONS? FEEL FREE TO ASK ANY–THING AT ALL.
WHY DO GHOSTS HAVE CLOTHES?
IF SOMEONE GIVES YOU A WEDGIE AT THE MOMENT YOU DIE, WILL YOU HAVE IT FOR ETERNITY?
Dilbert.com DilbertCartoonist@gmail.com
9-22-11 © 2011 Scott Adams, Inc./Dist. by Universal Uclick