# Designing Participatory Budgeting Mechanisms Grounded in Judgment Aggregation

Simon Rey, Ulle Endriss and Ronald de Haan

Institute for Logic, Language and Computation (ILLC)
University of Amsterdam
Amsterdam, The Netherlands

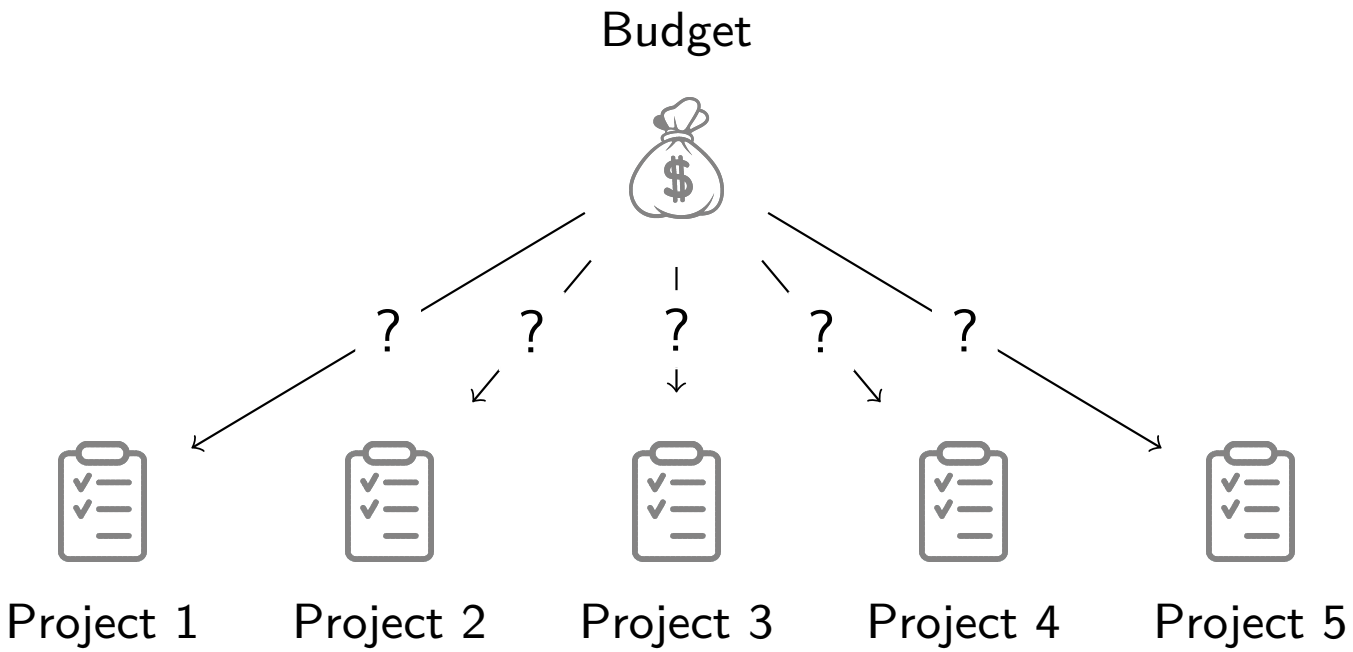
KR 2020

# Participatory Budgeting

Budget

[1] Cabannes "Participatory budgeting: A significant contribution to participatory democracy" (2004)
[2] Dias, Enriquez, and Julio *The Participatory Budgeting World Atlas* (2019)

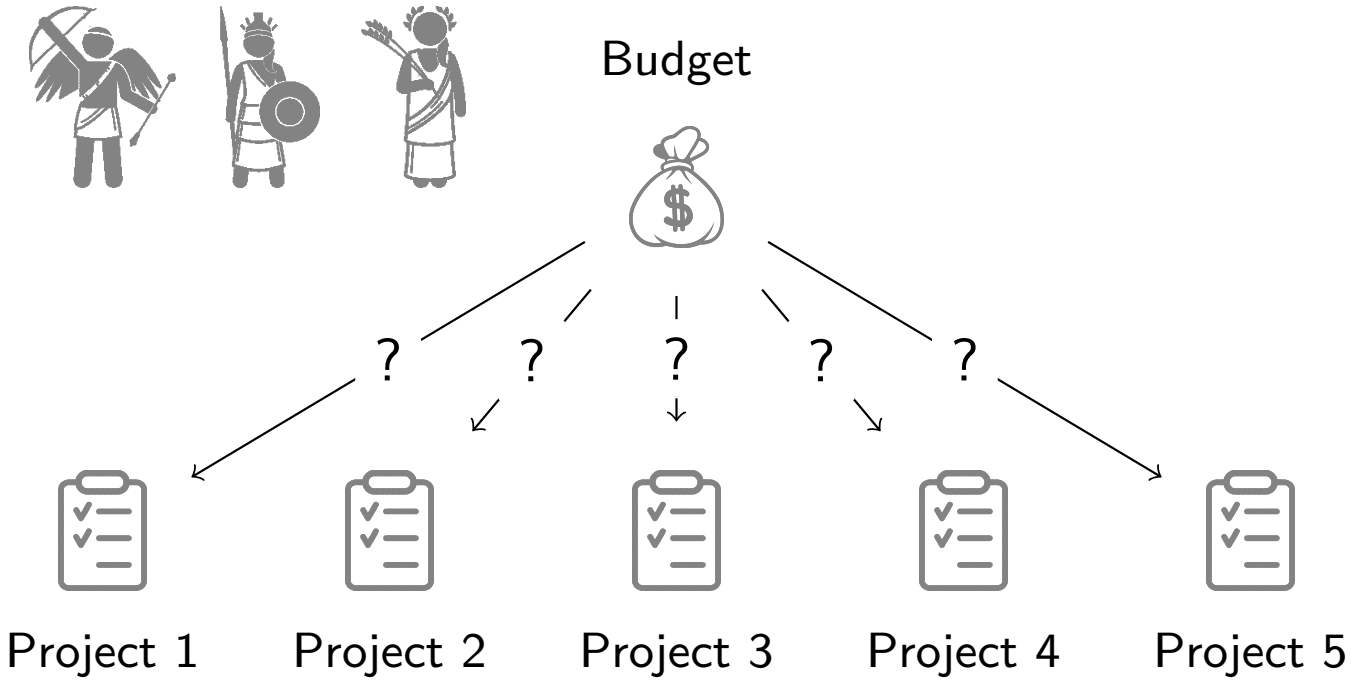
# Participatory Budgeting




---

[1] Cabannes "Participatory budgeting: A significant contribution to participatory democracy" (2004)
[2] Dias, Enriquez, and Julio *The Participatory Budgeting World Atlas* (2019)

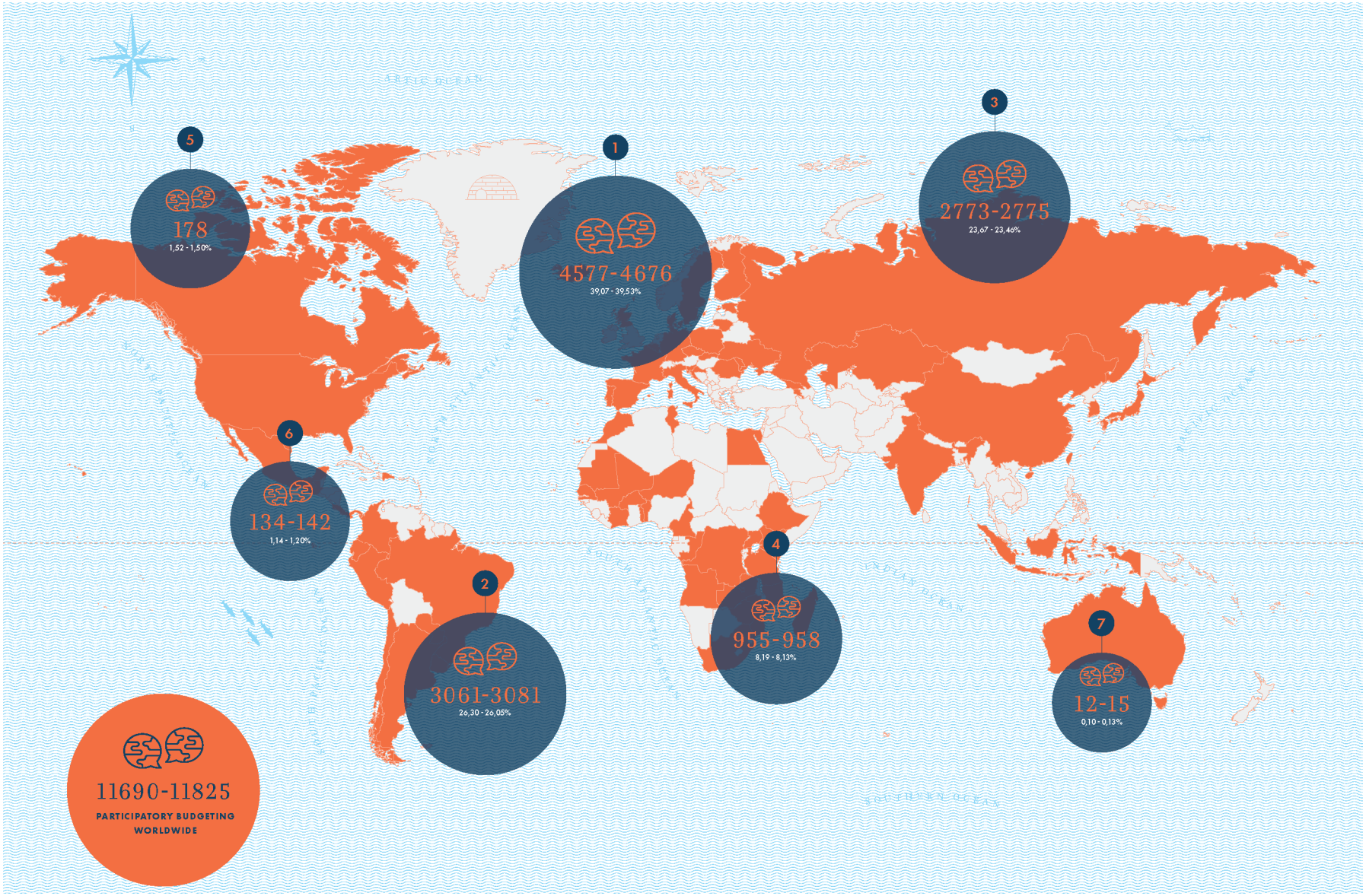
# Participatory Budgeting




---

[1] Cabannes "Participatory budgeting: A significant contribution to participatory democracy" (2004)
[2] Dias, Enriquez, and Julio *The Participatory Budgeting World Atlas* (2019)

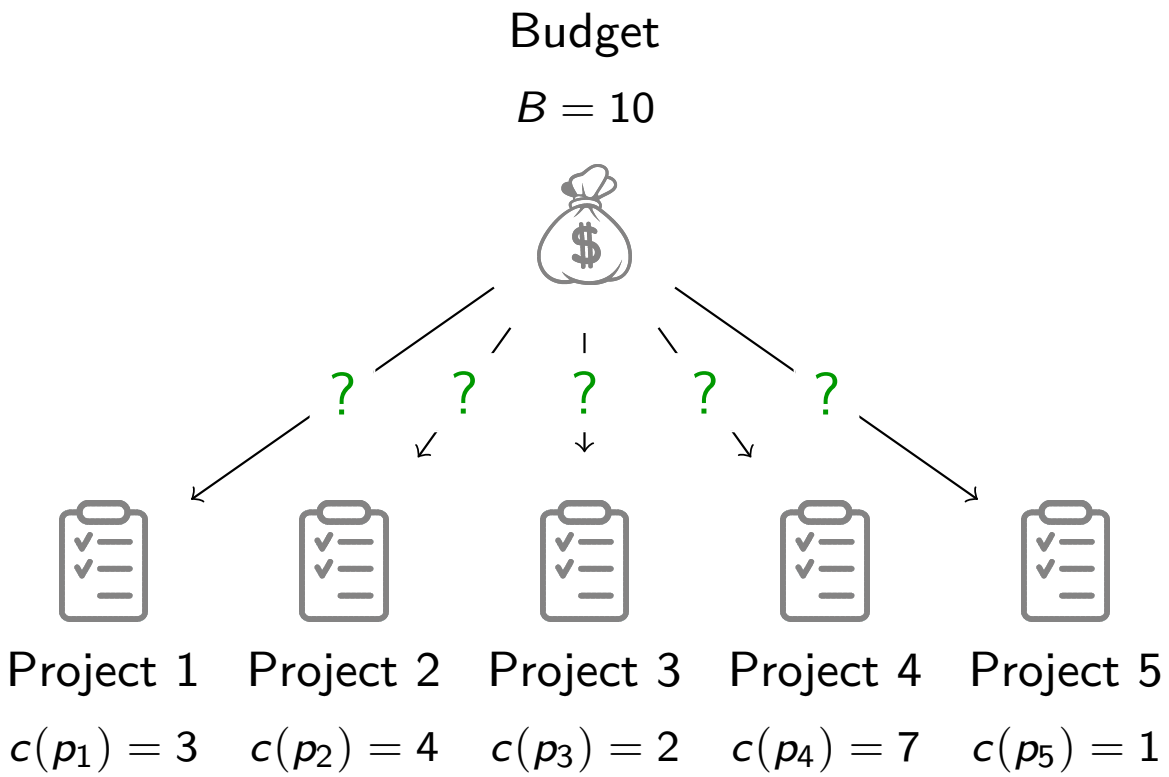
# PB around the world



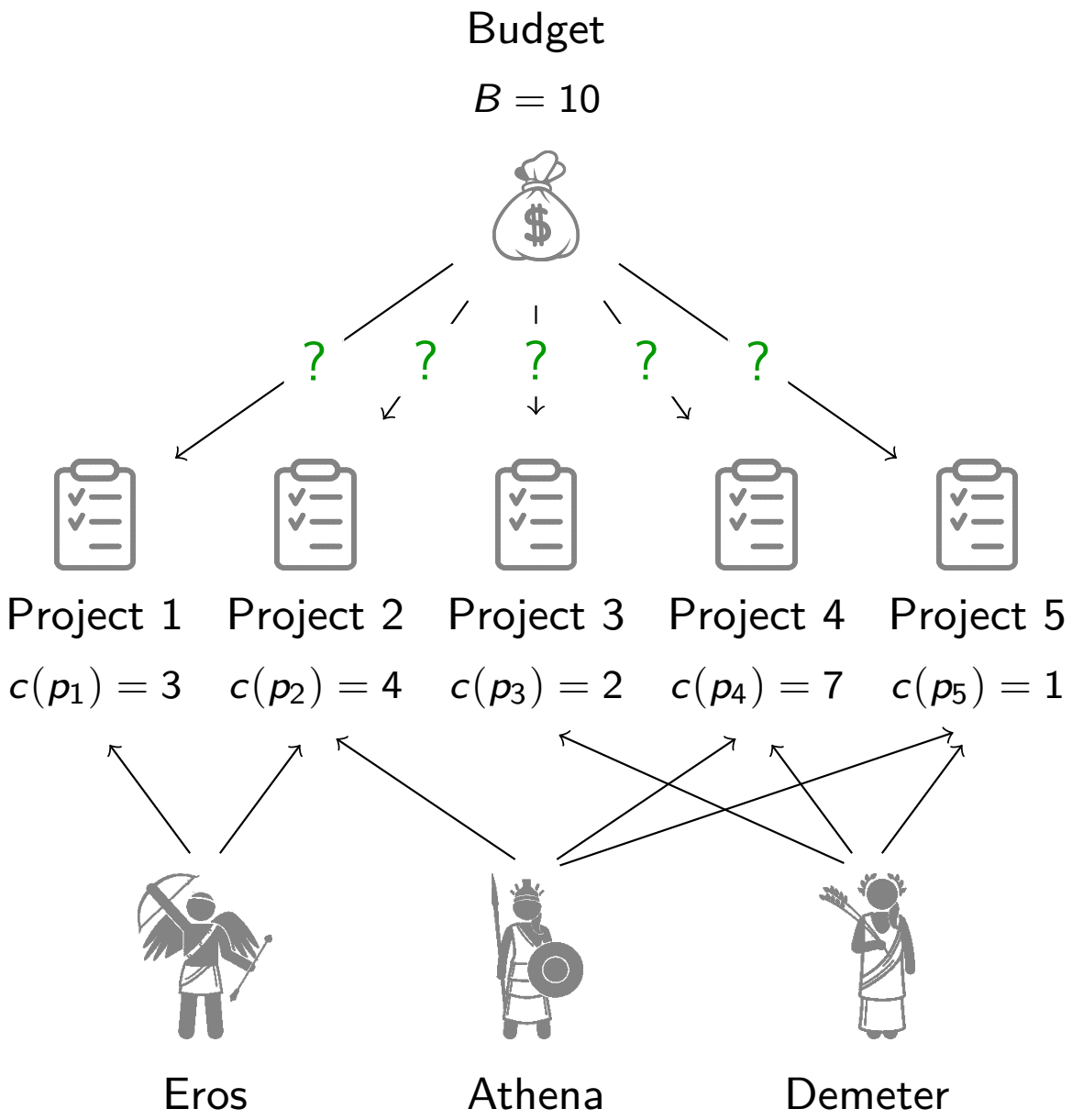
# The problem

| Project 1 | Project 2 | Project 3 | Project 4 | Project 5 |
|---|---|---|---|---|
| $c(p_1) = 3$ | $c(p_2) = 4$ | $c(p_3) = 2$ | $c(p_4) = 7$ | $c(p_5) = 1$ |

# The problem


Budget
$B = 10$
?
?
?
?
?
Project 1
$c(p_1) = 3$
Project 2
$c(p_2) = 4$
Project 3
$c(p_3) = 2$
Project 4
$c(p_4) = 7$
Project 5
$c(p_5) = 1$

# The problem


Budget
$B = 10$
?
?
?
?
?
Project 1
$c(p_1) = 3$
Project 2
$c(p_2) = 4$
Project 3
$c(p_3) = 2$
Project 4
$c(p_4) = 7$
Project 5
$c(p_5) = 1$
Eros
Athena
Demeter

# Current approach in ComSoC

Most of the works in the Computation Social Choice literature consider *Participatory Budgeting* as a generalization of *Multiwinner voting*:

- They both consider approval votes;
- When all projects are of cost 1, the two frameworks are indeed equivalent;
- PB would then be a multiwinner voting setting with costs on the alternatives.

 Each time a new feature is to be added, one needs to redefine *everything* to account for the new setting.

---

[3] Aziz and Shah "Participatory Budgeting: Models and Approaches" (2020)

# Our approach

Instead of generalizing already existing frameworks, can we use *highly expressive* ones while *keeping nice properties*?

# Judgment Aggregation

We will try to solve PB problems with the help of *Judgment Aggregation* (JA):

- Agents submit *approval ballots* over a set of issues;
- Using a *JA rule*, an aggregation of ballots is determined;
- The outcome must satisfy a *propositional formula* over the issues.

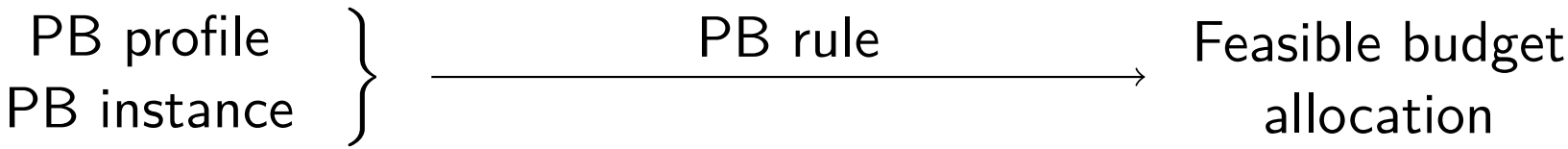
---

[4] Endriss "Judgment Aggregation" (2016)

# The Reduction

PB profile
PB instance

# The Reduction



# The Reduction

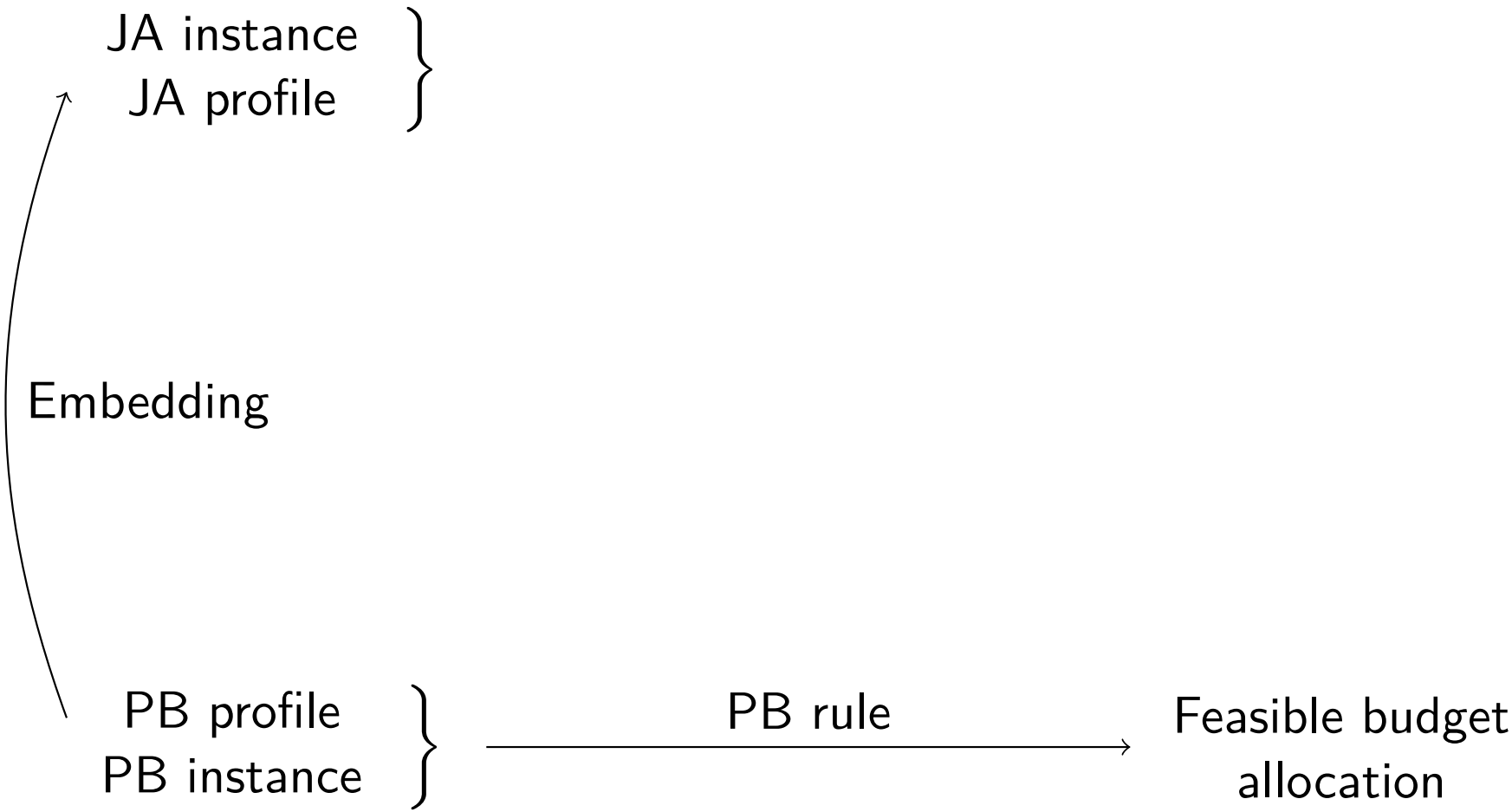
JA instance
JA profile

PB profile
PB instance

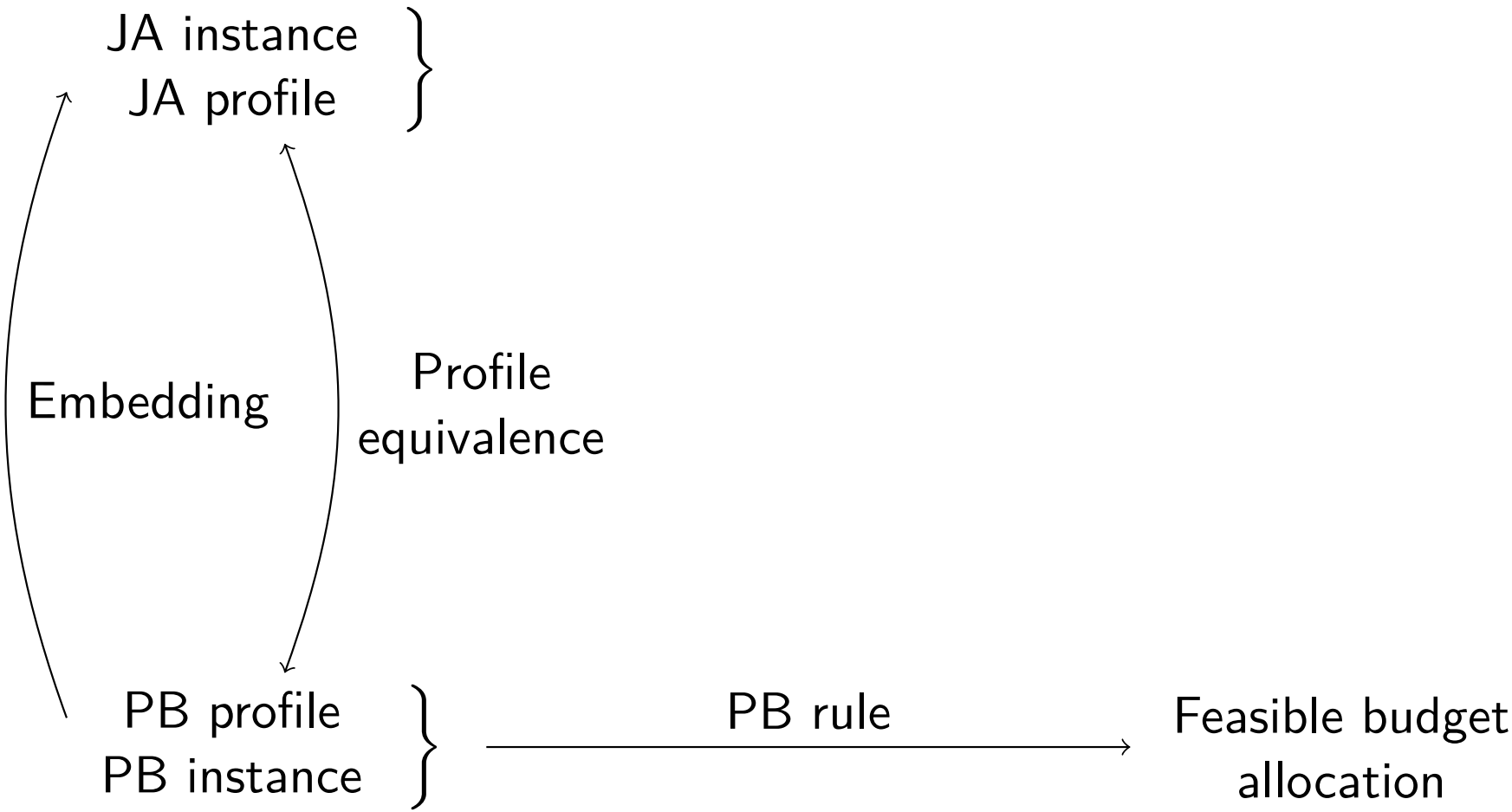
PB rule

Feasible budget allocation

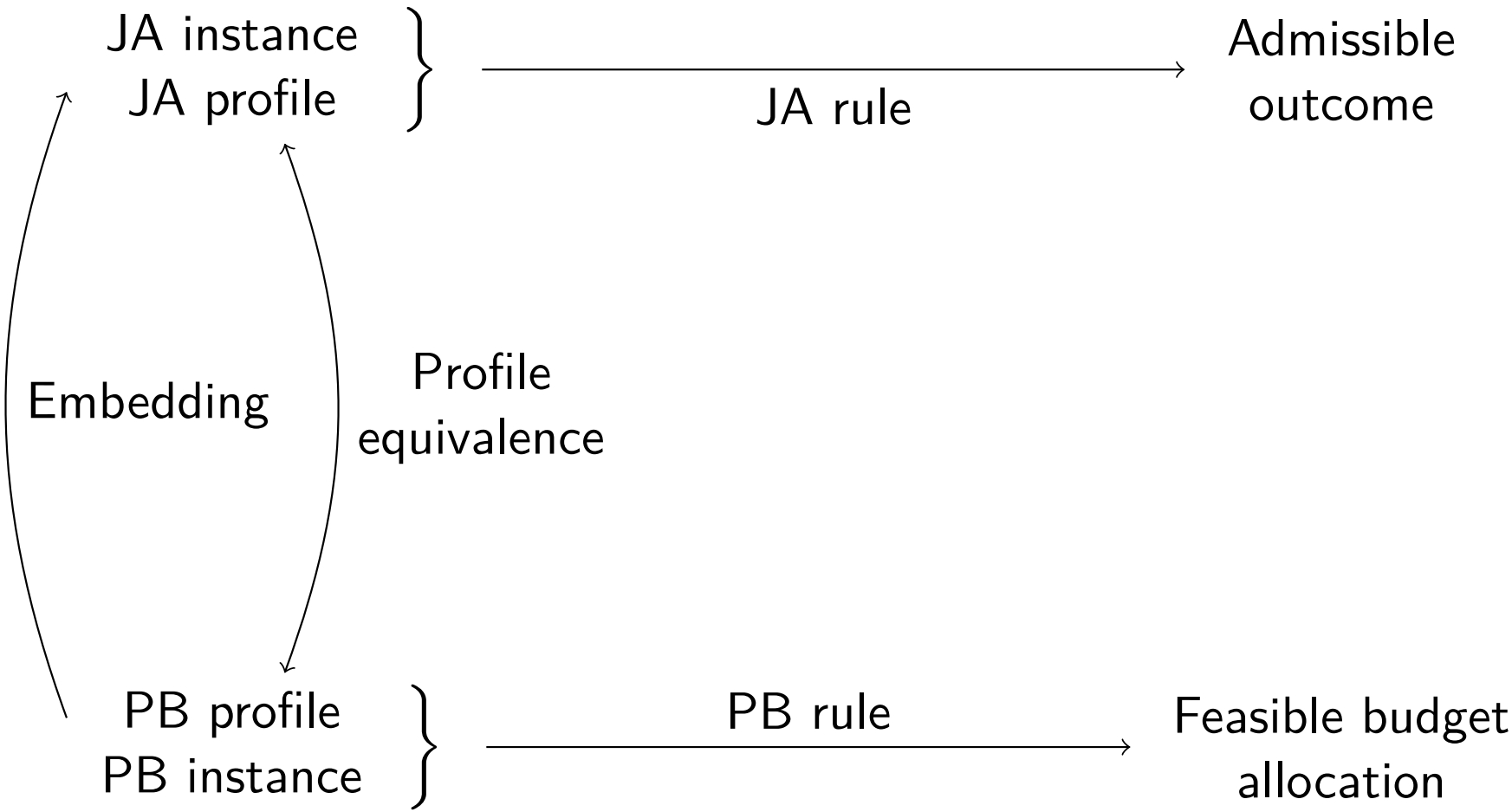
# The Reduction



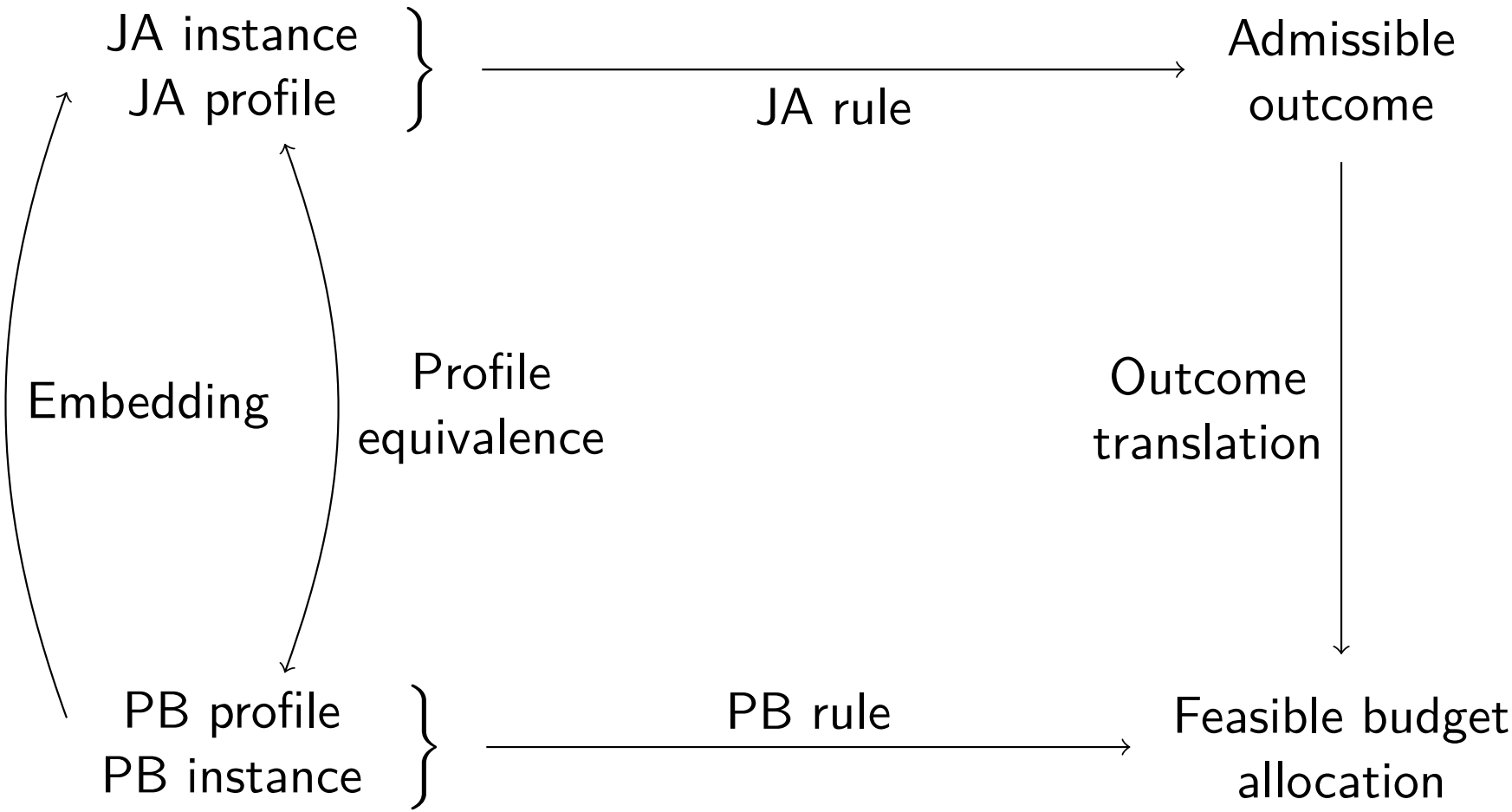
# The Reduction



# The Reduction



# The Reduction



# DNNF circuits

*Problem*: computing the outcome of JA rules usually is $\Theta_2^P$-complete.

↪ But not always, e.g. not if the input is represented as a *DNNF circuit*.

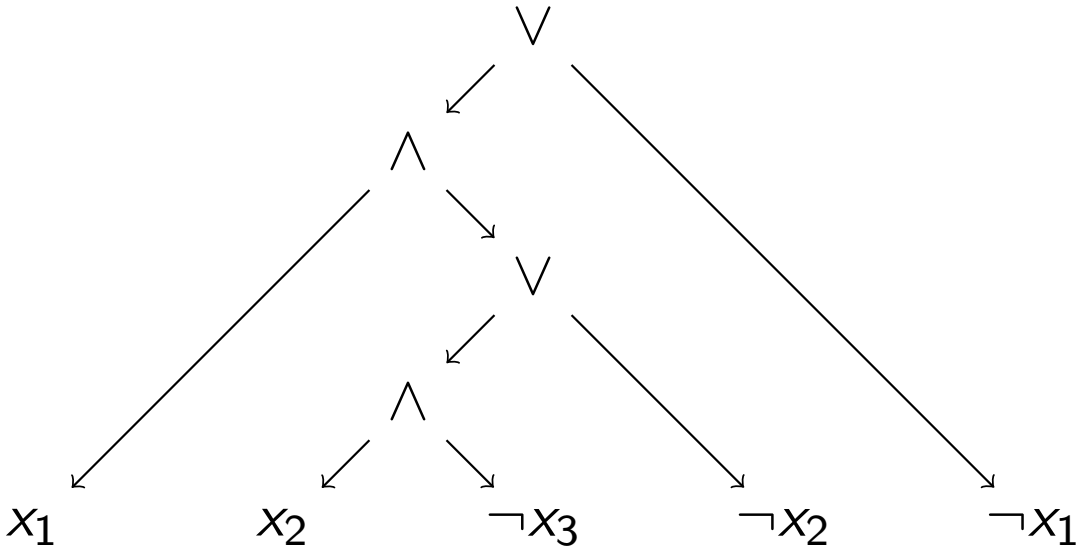
---

[5] De Haan "Hunting for Tractable Languages for Judgment Aggregation" (2018)

# DNNF circuits

*Problem*: computing the outcome of JA rules usually is $\Theta_2^P$-complete.

But not always, e.g. not if the input is represented as a *DNNF circuit*.



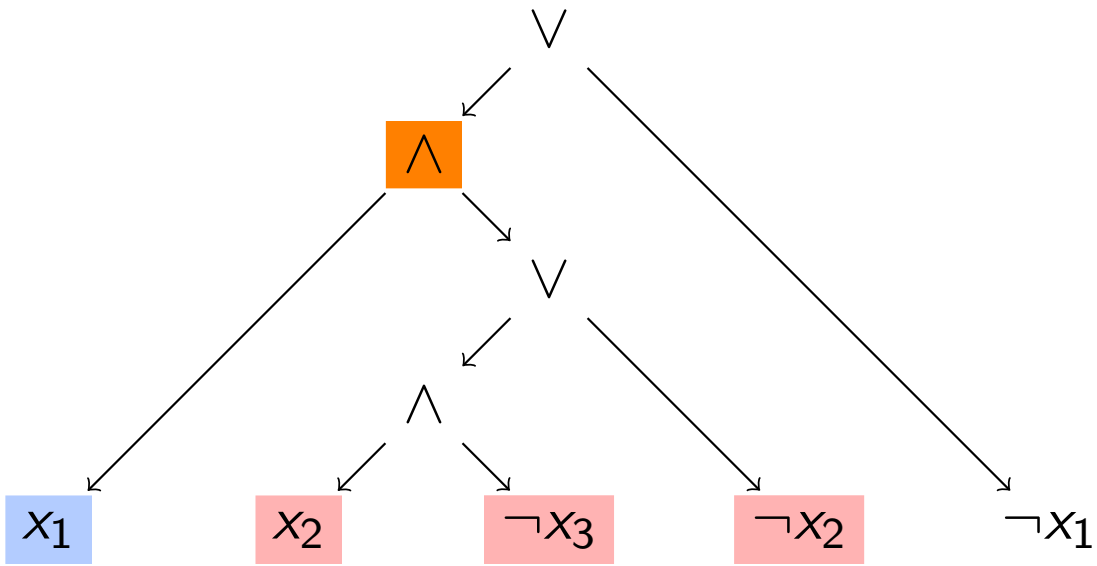

---

[5] De Haan "Hunting for Tractable Languages for Judgment Aggregation" (2018)

# DNNF circuits

*Problem*: computing the outcome of JA rules usually is $\Theta_2^P$-complete.

But not always, e.g. not if the input is represented as a *DNNF circuit*.



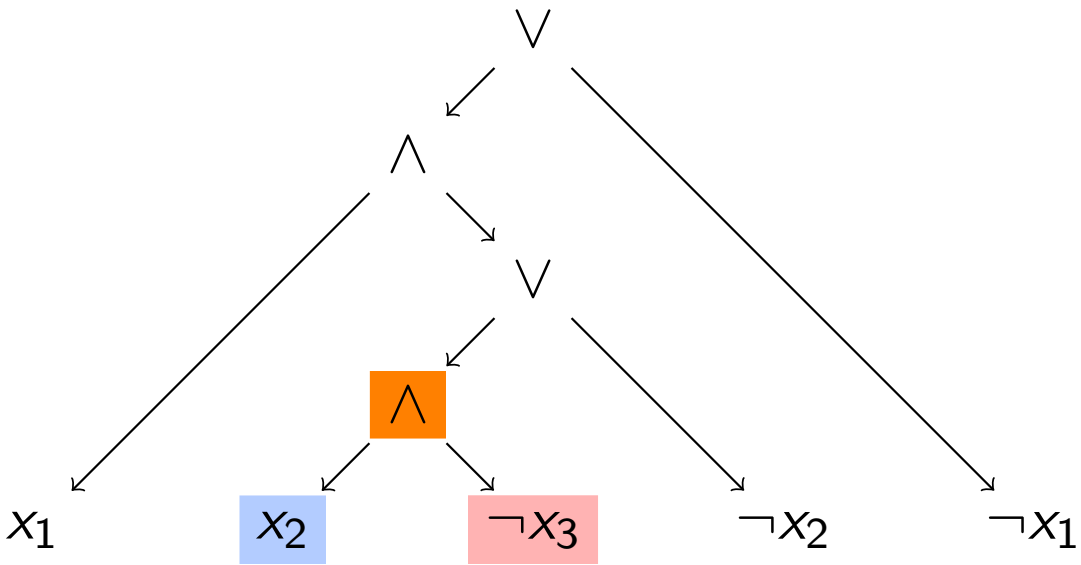

---

[5] De Haan "Hunting for Tractable Languages for Judgment Aggregation" (2018)

# DNNF circuits

*Problem*: computing the outcome of JA rules usually is $\Theta_2^P$-complete.

But not always, e.g. not if the input is represented as a *DNNF circuit*.




---

[5] De Haan "Hunting for Tractable Languages for Judgment Aggregation" (2018)

# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$

# Encoding Budgeting Problems in DNNF

$B = 2$

$N(0, 1)$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$

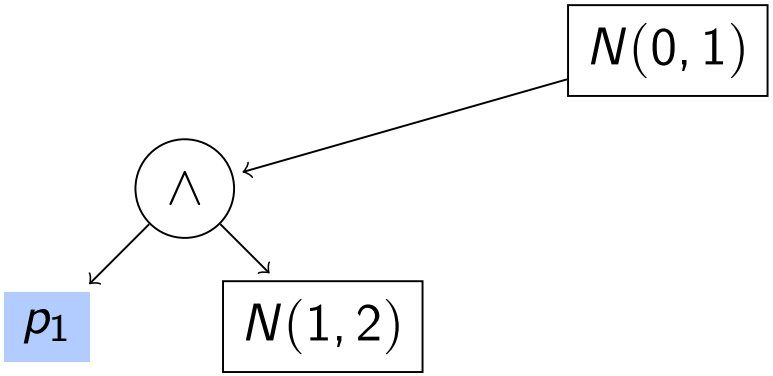
# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$



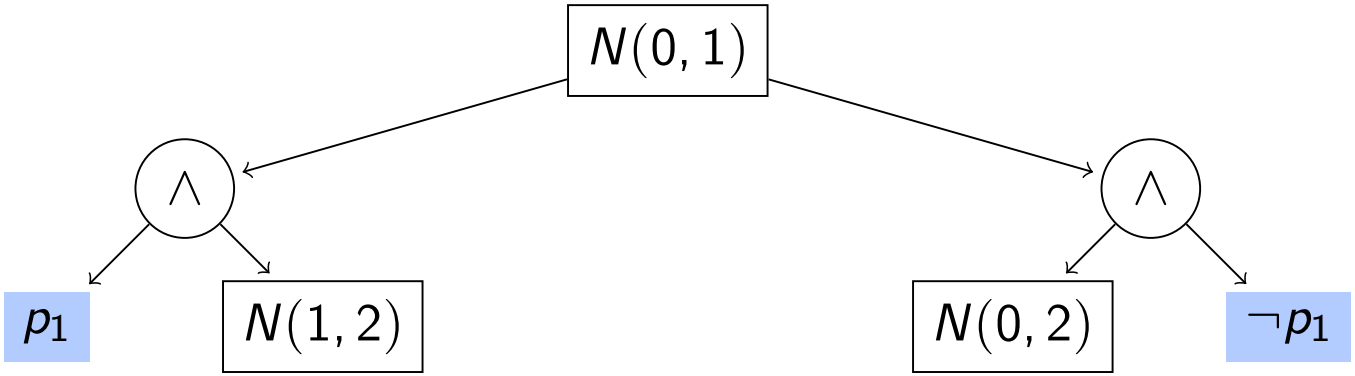
# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$



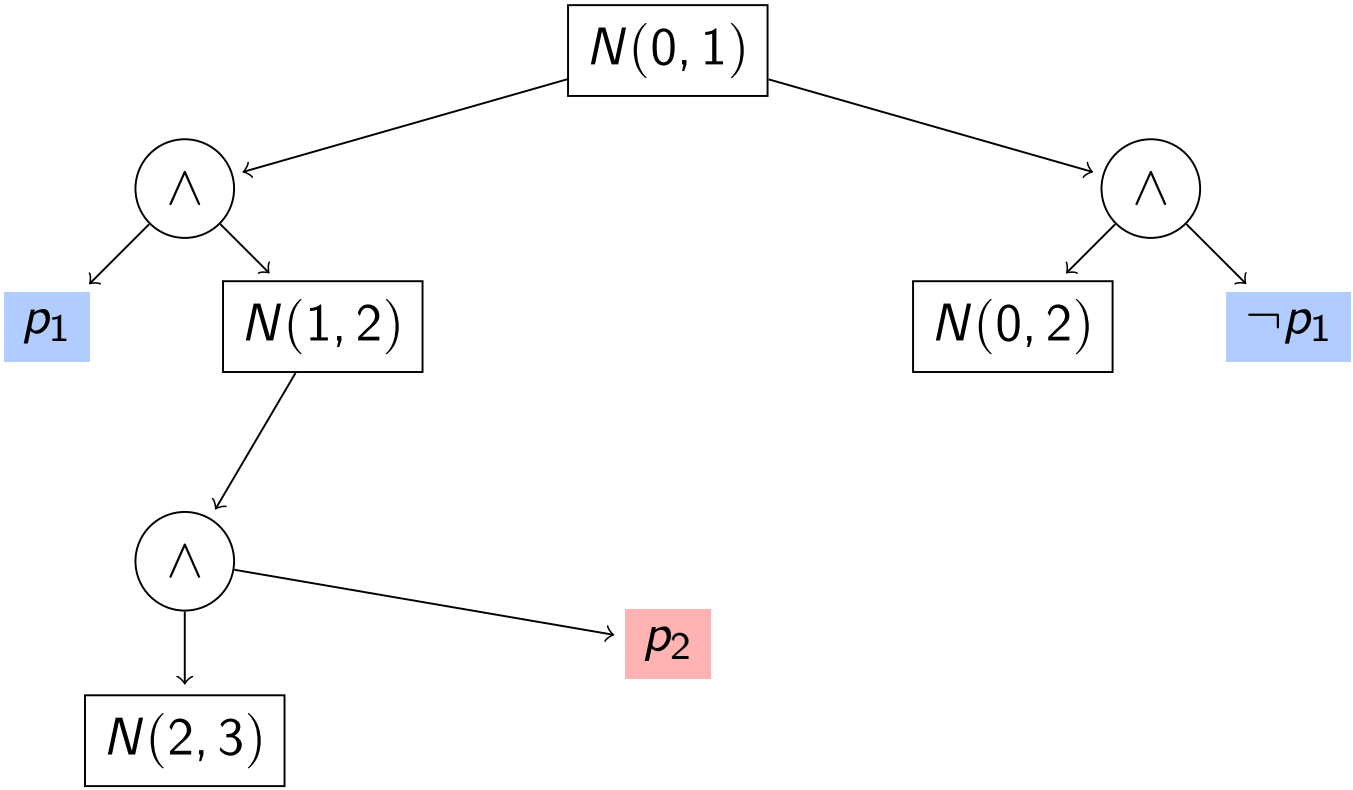
# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$



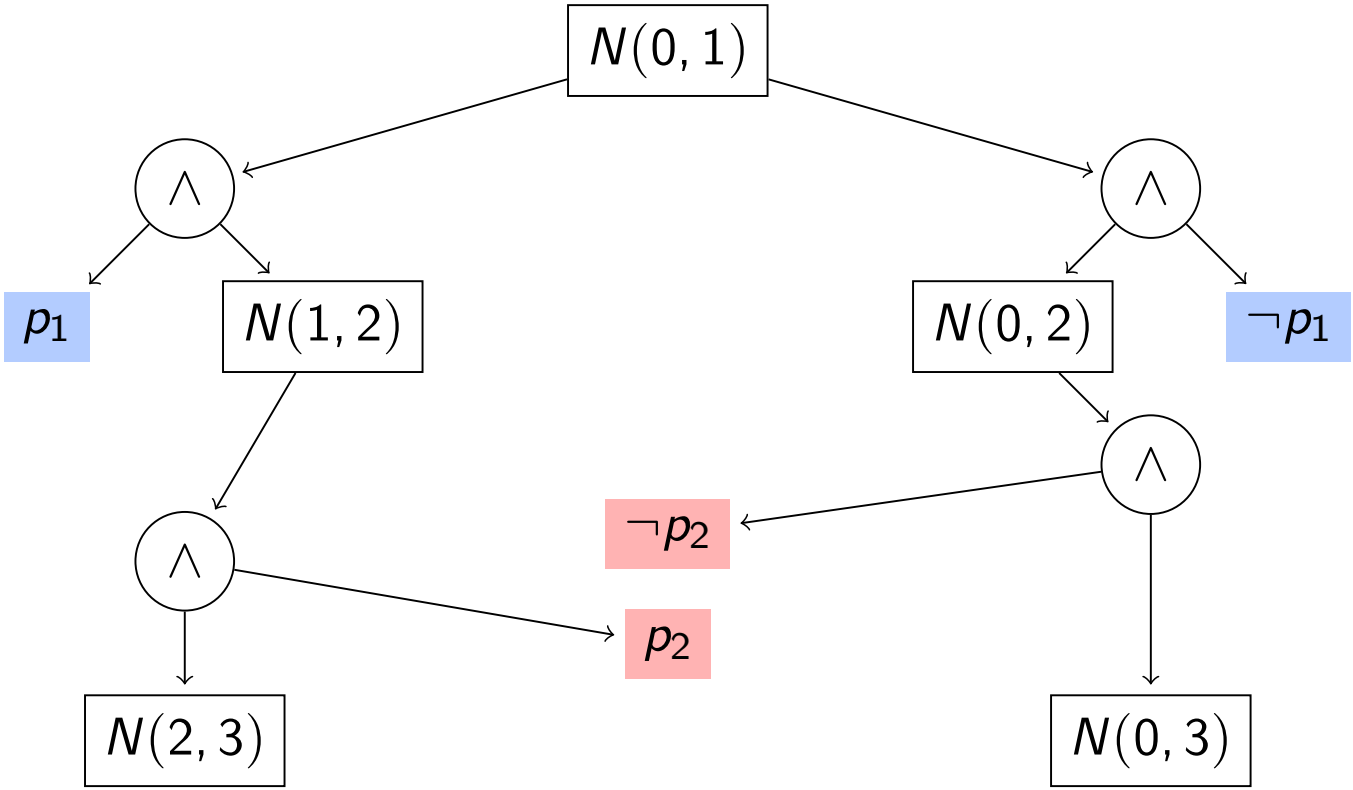
# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$



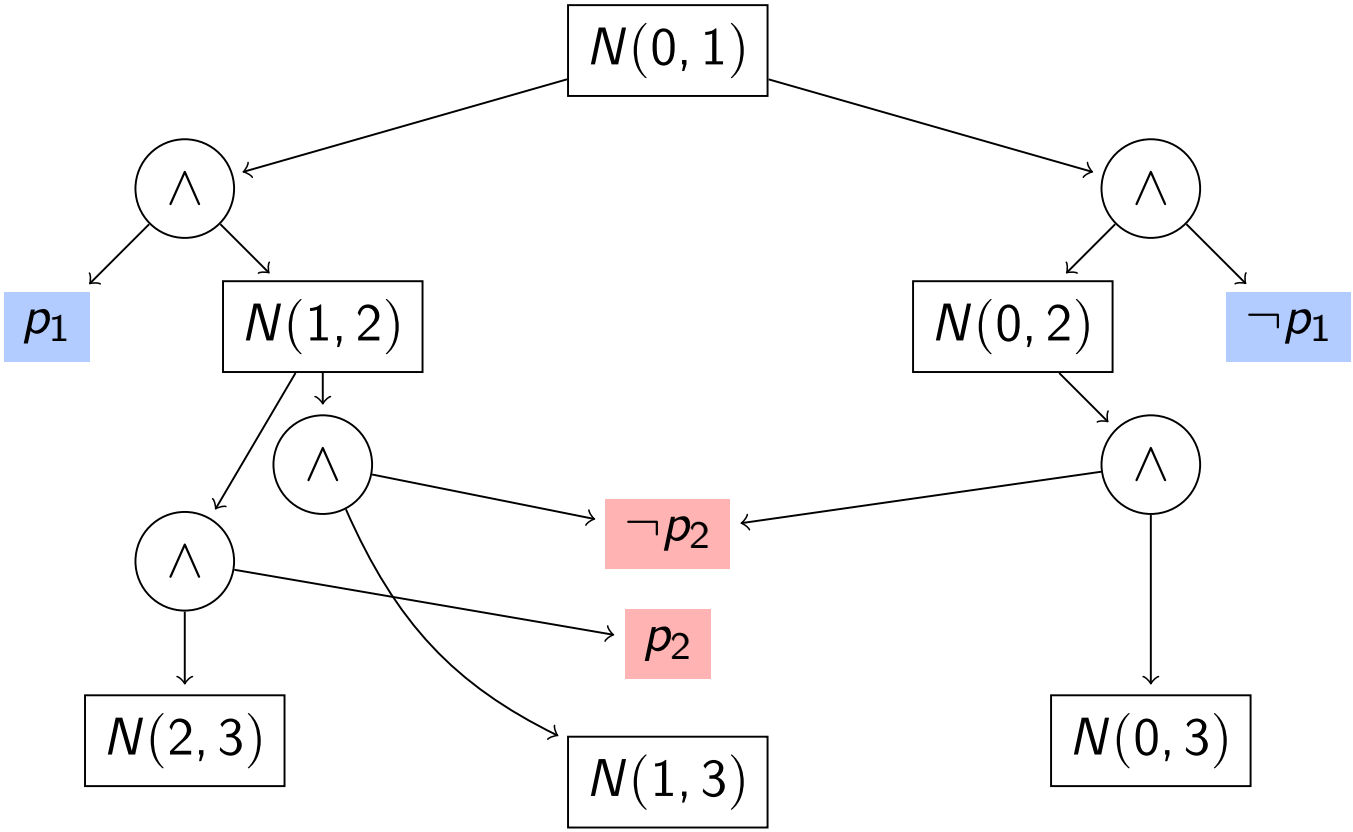
# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$



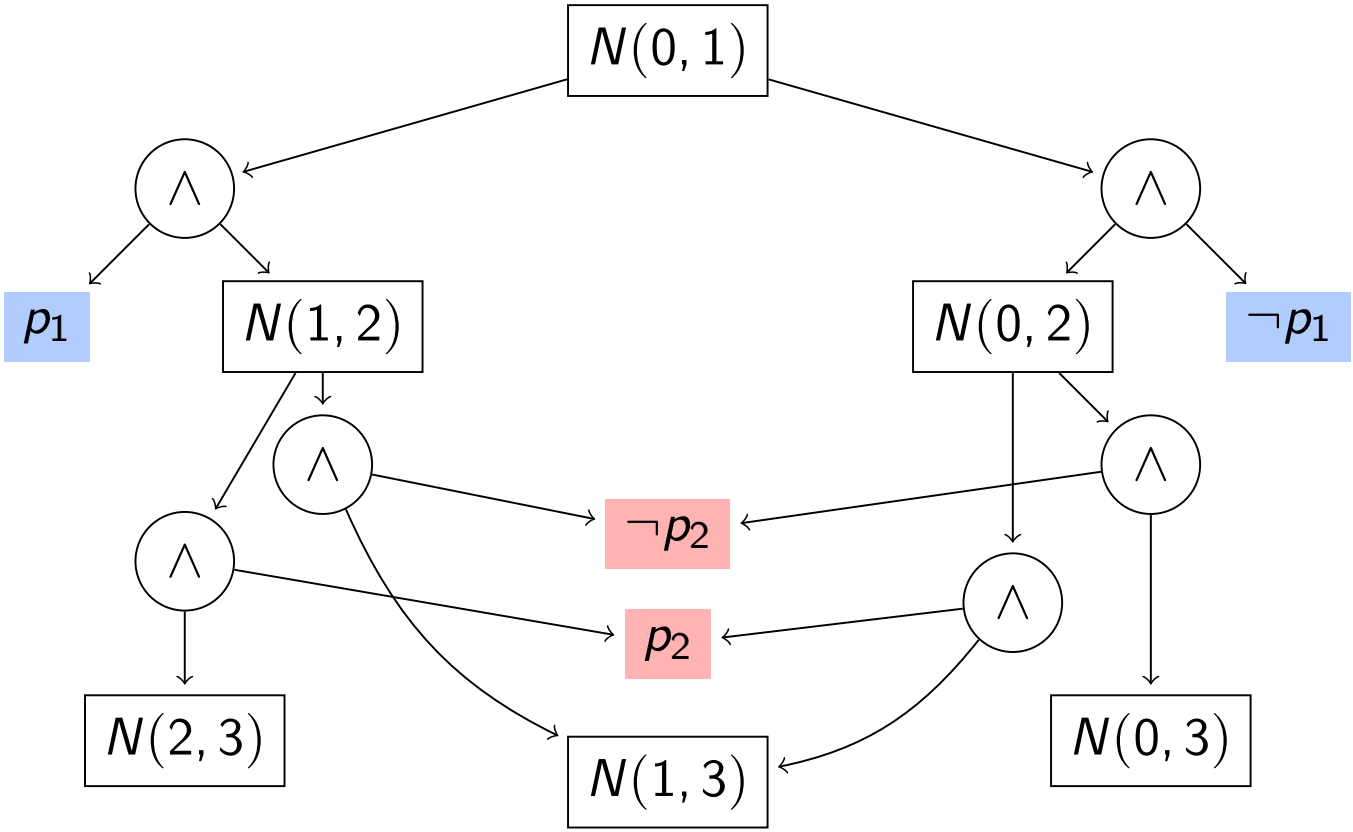
# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$



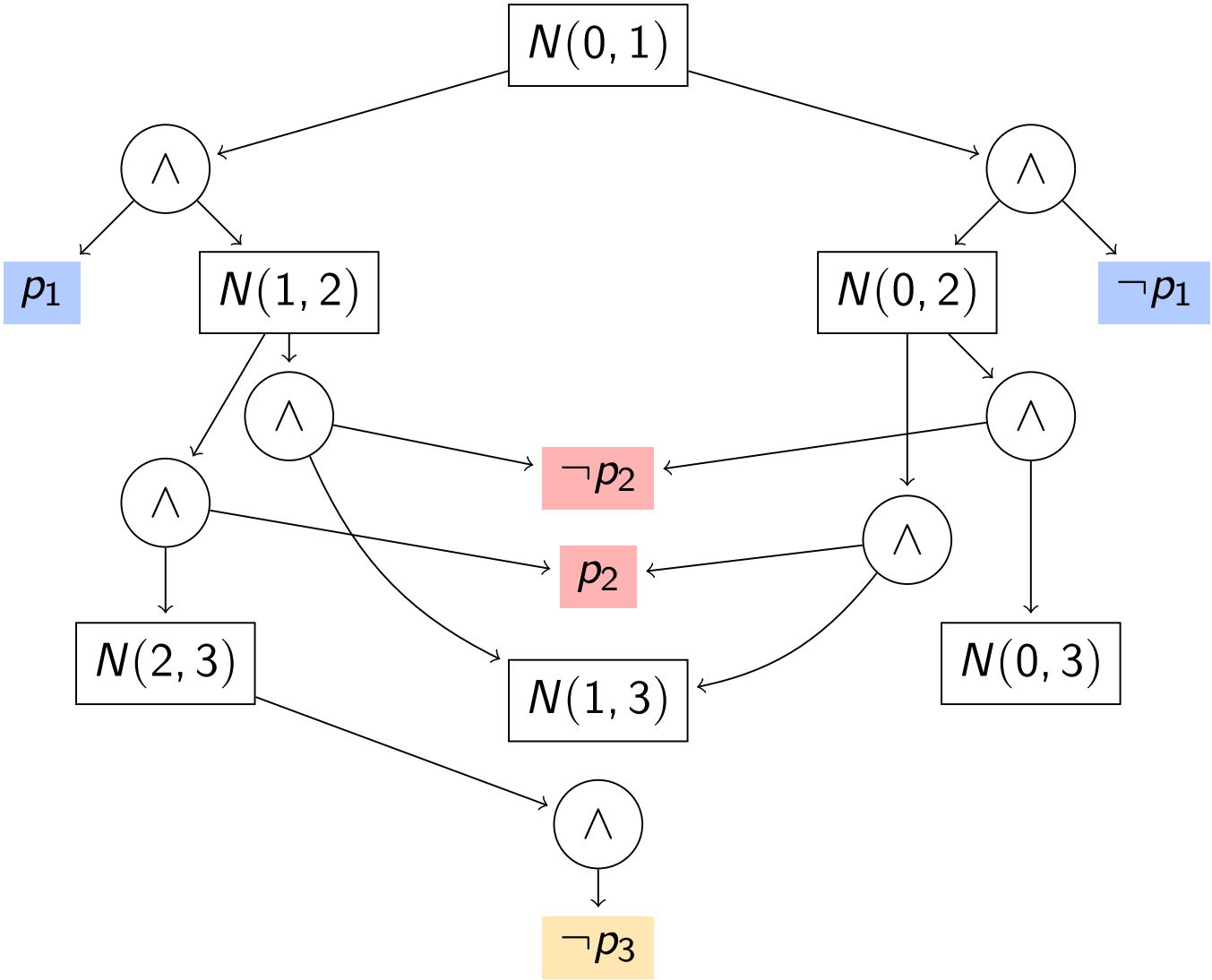
# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$



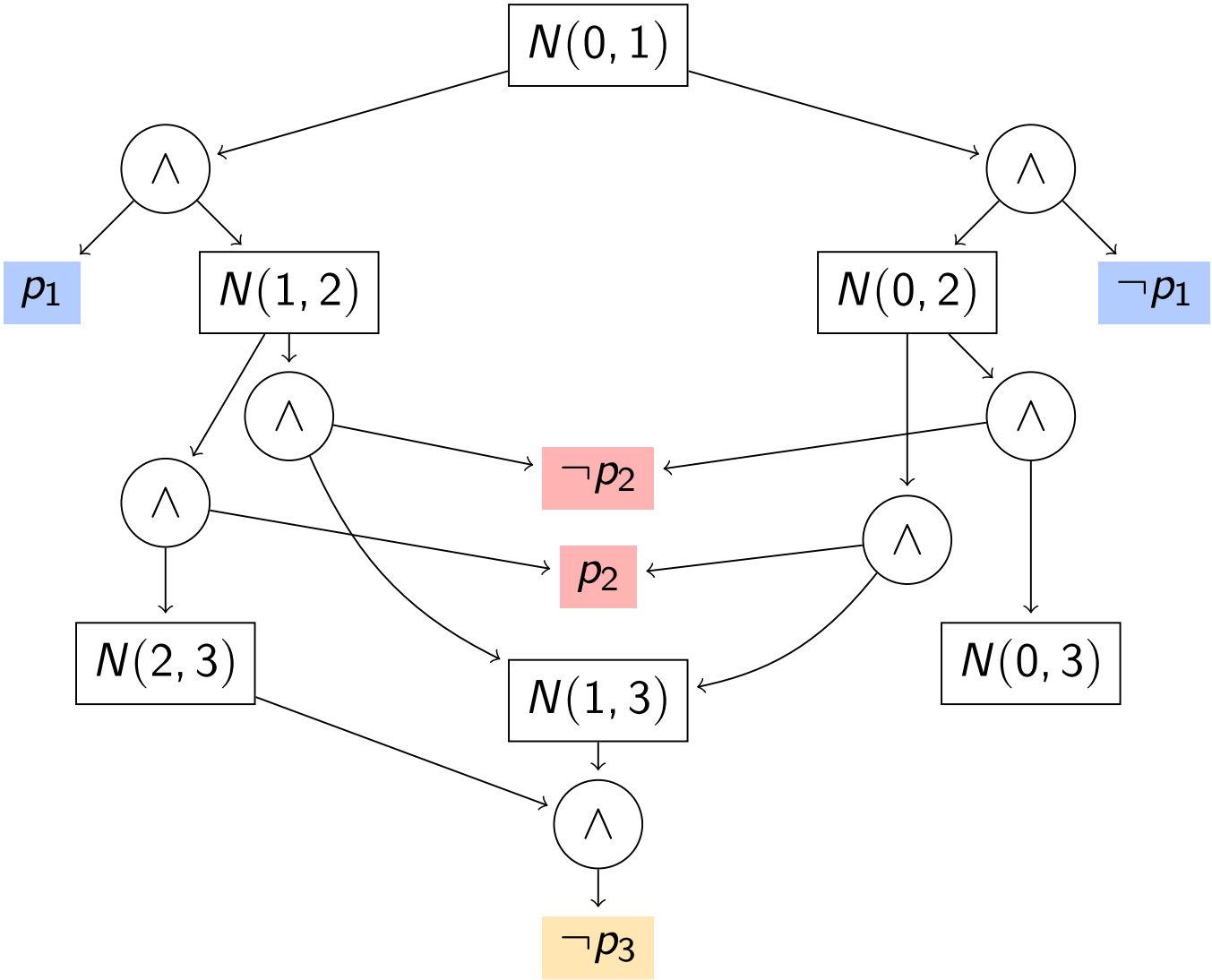
# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$



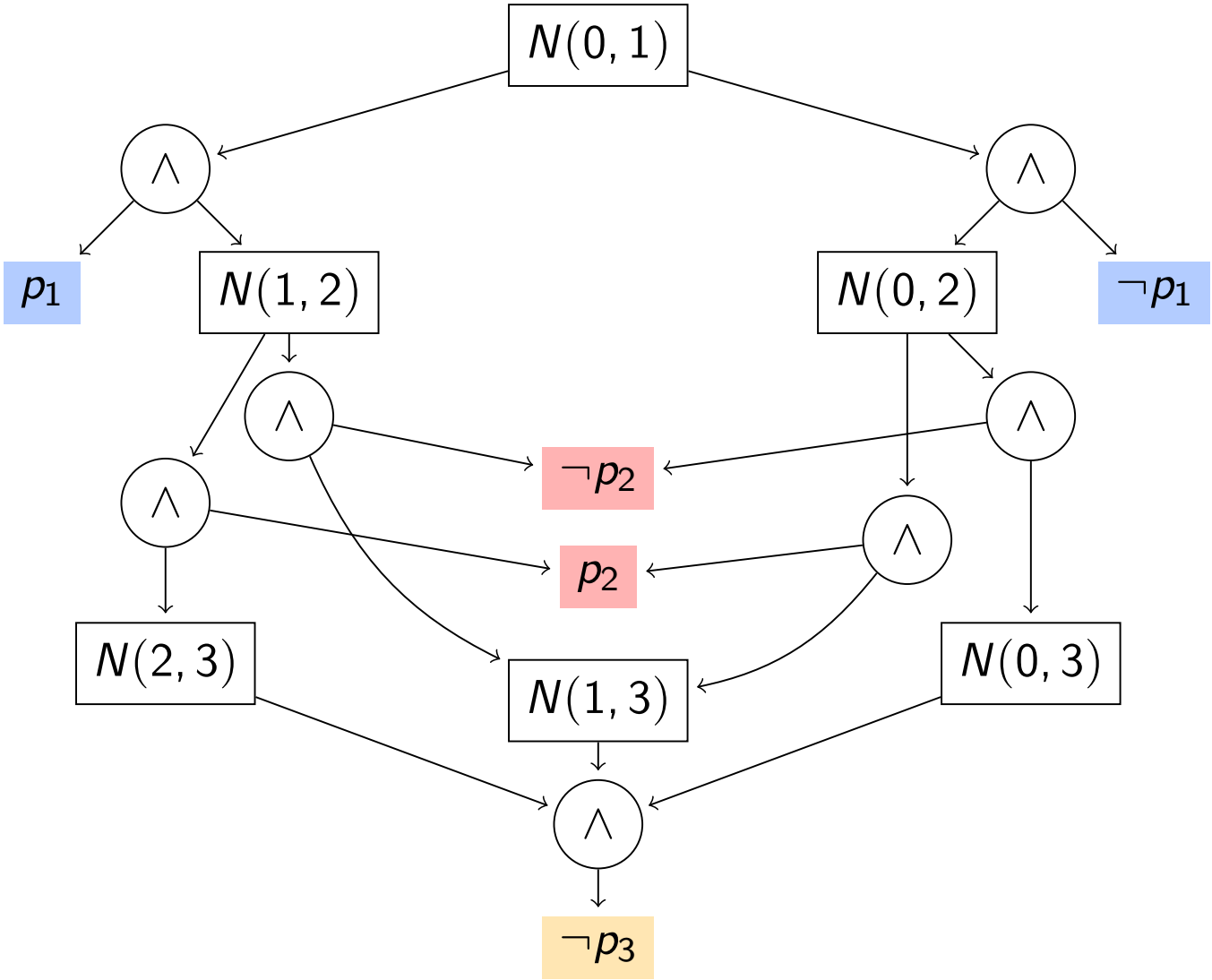
# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$



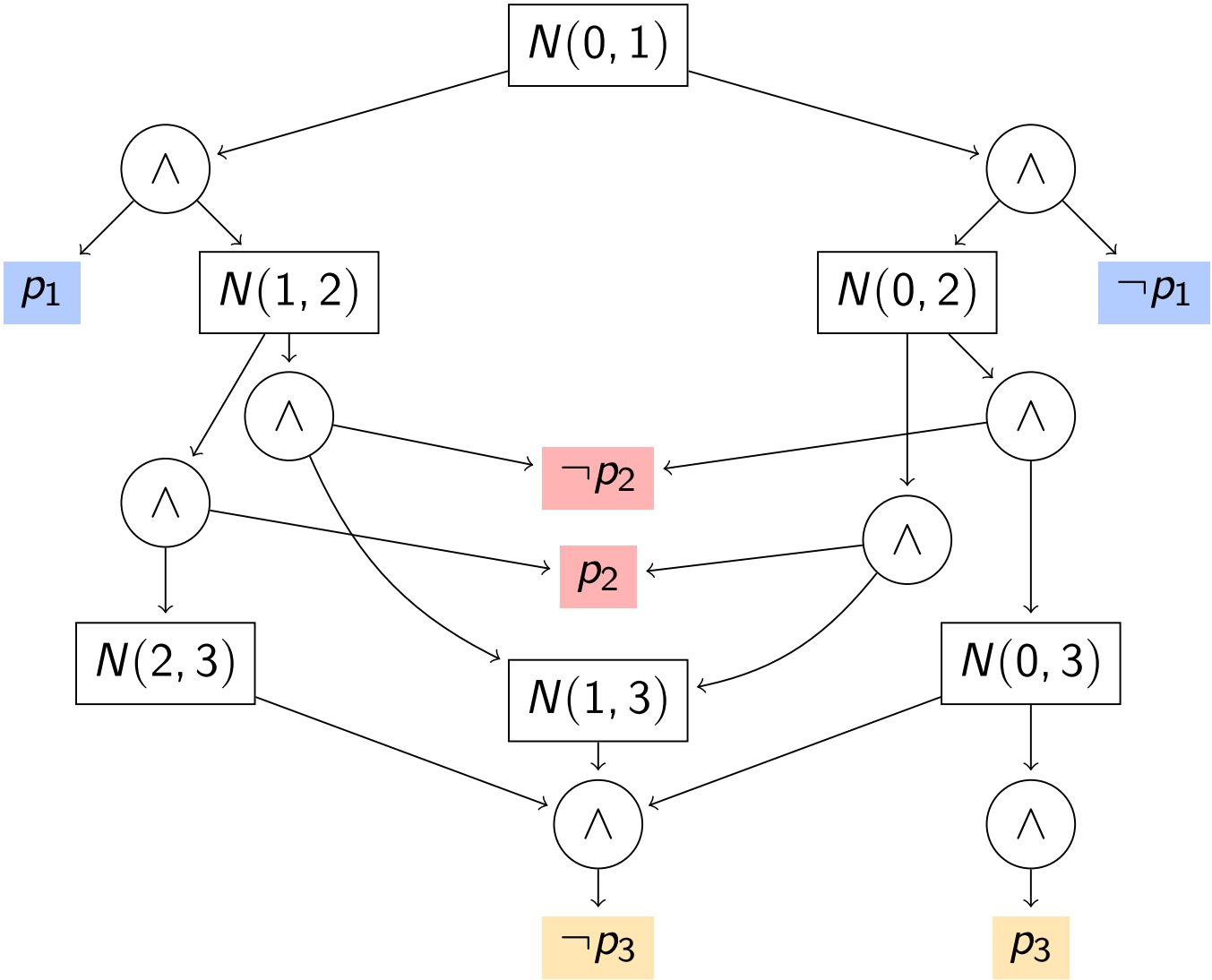
# Encoding Budgeting Problems in DNNF

$B = 2$

$c(p_1) = 1$

$c(p_2) = 1$

$c(p_3) = 2$



# Adding dependencies and/or quotas over the projects

THEOREM:

Computing a feasible budget allocation with dependencies and/or quotas is NP-complete.

# Adding dependencies and/or quotas over the projects

THEOREM:

Computing a feasible budget allocation with dependencies and/or quotas is NP-complete.

THEOREM:

PB with dependencies and/or quotas can be embedded into JA via parametrized embeddings.

# Conclusion

We presented:

- *Embeddings* from PB to JA;
- That allowed for great *expressivity*;
- And analyzed actual *performance* of JA rules for PB.

 This work shows interesting interconnections between Social Choice and Knowledge Representation.

Any questions?

s.j.rey@uva.nl
u.endriss@uva.nl
r.dehaan@uva.nl