Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it
Bonn-Aachen
International Center for
Information Technology

Institute for AI and
Autonomous Systems

Relational Learning

A Short Introduction

Dr. Alex Mitrevski
Master of Autonomous Systems

Structure

- Relational learning preliminaries
- Relational learning methods

Robotics and Autonomous Systems 161 (2023) 104350

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

ELSEVIER

A hybrid skill parameterisation model combining symbolic and subsymbolic elements for introspective robots

Alex Mitrevski [a,b,*], Paul G. Plöger [a], Gerhard Lakemeyer [b]

Alex Mitrevski

Improving the Manipulation Skills of Service Robots by Refining Action Representations

Skill generalisation and experience acquisition for predicting and avoiding execution failures = Fähigkeitsgeneralisierung und Erfahrungserwerb zum Vorhersagen und Vermeiden von Ausführungsfehlern

Mitrevski, Aleksandar (Alex)

Relational Learning Preliminaries

Relational Learning Preliminaries

- In our first lectures, we defined a skill as $\mathcal{S} = (\mathcal{S}_{\mathcal{I}}, \mathcal{S}_{\mathcal{T}}, \pi)$, where $\mathcal{S}_{\mathcal{I}}$ are skill preconditions and $\mathcal{S}_{\mathcal{T}}$ are skill termination conditions, but we did not discuss how those can be acquired — this is our general objective for today

Relational Learning Preliminaries

- In our first lectures, we defined a skill as $\mathcal{S} = (\mathcal{S}_{\mathcal{I}}, \mathcal{S}_{\mathcal{T}}, \pi)$, where $\mathcal{S}_{\mathcal{I}}$ are skill preconditions and $\mathcal{S}_{\mathcal{T}}$ are skill termination conditions, but we did not discuss how those can be acquired — this is our general objective for today
- Execution preconditions and effects can be acquired through **relational learning**, whose general objective is to learn rules (often interpreted as conjunctions of predicates) that are **consistent with a given set of observations**

Relational Learning Preliminaries

- In our first lectures, we defined a skill as $\mathcal{S} = (\mathcal{S}_\mathcal{I}, \mathcal{S}_\mathcal{T}, \pi)$, where $\mathcal{S}_\mathcal{I}$ are skill preconditions and $\mathcal{S}_\mathcal{T}$ are skill termination conditions, but we did not discuss how those can be acquired — this is our general objective for today
- Execution preconditions and effects can be acquired through **relational learning**, whose general objective is to learn rules (often interpreted as conjunctions of predicates) that are **consistent with a given set of observations**
- For simplicity, the relations from which rules are learned are often assumed to be given; however, relational learning also encompasses the process of **learning relations as such**

Relational Learning Preliminaries

- In our first lectures, we defined a skill as $\mathcal{S} = (\mathcal{S}_\mathcal{I}, \mathcal{S}_\mathcal{T}, \pi)$, where $\mathcal{S}_\mathcal{I}$ are skill preconditions and $\mathcal{S}_\mathcal{T}$ are skill termination conditions, but we did not discuss how those can be acquired — this is our general objective for today
- Execution preconditions and effects can be acquired through **relational learning**, whose general objective is to learn rules (often interpreted as conjunctions of predicates) that are **consistent with a given set of observations**
- For simplicity, the relations from which rules are learned are often assumed to be given; however, relational learning also encompasses the process of **learning relations as such**
- Beyond skill models that can be used for autonomous planning, relational learning has another use: it acquires **explainable models of execution**

Relational Learning Preliminaries

- In our first lectures, we defined a skill as $\mathcal{S} = (\mathcal{S}_\mathcal{I}, \mathcal{S}_\mathcal{T}, \pi)$, where $\mathcal{S}_\mathcal{I}$ are skill preconditions and $\mathcal{S}_\mathcal{T}$ are skill termination conditions, but we did not discuss how those can be acquired — this is our general objective for today
- Execution preconditions and effects can be acquired through **relational learning**, whose general objective is to learn rules (often interpreted as conjunctions of predicates) that are **consistent with a given set of observations**
- For simplicity, the relations from which rules are learned are often assumed to be given; however, relational learning also encompasses the process of **learning relations as such**
- Beyond skill models that can be used for autonomous planning, relational learning has another use: it acquires **explainable models of execution**

Relational learning is a problem of extracting qualitative rules from observations, or learning models of relations themselves

Relational Learning and Planning Domains

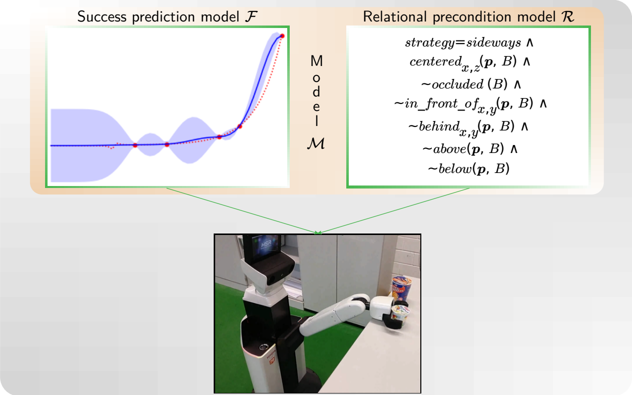
- A traditional use of relational learning is for **learning planning domains from observed plan traces** so that there is no need for writing domains manually

The snippet is based on an example used in the presentation of A. Mitrevski et al., "On the Diagnosability of Actions Performed by Contemporary Robotic Systems," in *31th Int. Workshop Principles of Diagnosis (DX)*, 2020.

```
(:action Pick
    :parameters (?Object - Object ?Table - Table ?Robot - Robot ?Waypoint -
           Waypoint)
    :precondition (and
         (robotAt ?Robot ?Waypoint)
         (tableAt ?Table ?Waypoint)
         (on ?Object ?Table)
         (emptyGripper ?Robot)
    )
    :effect (and
         (not (on ?Object ?Table))
         (not (emptyGripper ?Robot))
         (holding ?Robot ?Object)
    )
)
```

Relational Learning and Execution

- Relational models can also be used to describe execution rules, but specifying those manually is not a generalisable approach; **relational learning can also be used for learning execution models**



A. Mitrevski, P. G. Plöger, and G. Lakemeyer, "A Hybrid Skill Parameterisation Model Combining Symbolic and Subsymbolic Elements for Introspective Robots," *Robotics and Autonomous Systems*, vol. 161, p. 104350:1–22, Mar. 2023.

Relational Learning Tasks

Skill precondition learning

The objective here is to learn a model of skill preconditions, typically as a **conjunction of positive literals**

Relational Learning Tasks

Skill precondition learning

The objective here is to learn a model of skill preconditions, typically as a **conjunction of positive literals**

Learning effects / termination conditions

In this task, the objective is to acquire skill effects domain, typically as a **conjunction of positive and negative literals**

Relational Learning Tasks

Skill precondition learning

The objective here is to learn a model of skill preconditions, typically as a **conjunction of positive literals**

Learning effects / termination conditions

In this task, the objective is to acquire skill effects domain, typically as a **conjunction of positive and negative literals**

Learning execution rules

Learning can also be used to describe relational execution rules, including rules about the temporal evolution of a task

Relational Learning Tasks

Skill precondition learning

The objective here is to learn a model of skill preconditions, typically as a **conjunction of positive literals**

Learning execution rules

Learning can also be used to describe relational execution rules, including rules about the temporal evolution of a task

Learning effects / termination conditions

In this task, the objective is to acquire skill effects domain, typically as a **conjunction of positive and negative literals**

Symbol learning

The other three tasks assume that a set of relevant relations is given and these are used to learn appropriate rules; in symbol learning, the objective is to **learn the relations themselves**

Relational Learning Methods

Type of Relational Learning Tasks

- There are two general classes of relational learning problem, depending on whether candidate relations are given:

Type of Relational Learning Tasks

- There are two general classes of relational learning problem, depending on whether candidate relations are given:

Known relations

In the case of a known set of relations, experiences are used to identify relevant relations from the set; this is a **feature extraction task**

Type of Relational Learning Tasks

- There are two general classes of relational learning problem, depending on whether candidate relations are given:

Known relations

In the case of a known set of relations, experiences are used to identify relevant relations from the set; this is a **feature extraction task**

Unknown relations

If the relations are unknown, experiences need to be used to find new relations; this is a **symbol learning task**

Type of Relational Learning Tasks

- There are two general classes of relational learning problem, depending on whether candidate relations are given:

Known relations

In the case of a known set of relations, experiences are used to identify relevant relations from the set; this is a **feature extraction task**

Unknown relations

If the relations are unknown, experiences need to be used to find new relations; this is a **symbol learning task**

- In this lecture, we will mostly focus on the case of learning with known relations as it is a better studied problem; symbol learning will be discussed only briefly

Learning Objective: Known Relations

- Let us suppose that we have **a set P of candidate relations**, where $r \in P$ is defined as either $r : S \rightarrow \mathbb{Z}$ or $r : S \rightarrow \mathbb{R}$, with S a state representation
  - E.g. P can be a set such as $\{leftOf(X, Y), rightOf(X, Y), inFrontOf(X, Y), behind(X, Y)\}$
  - Typically, the relations we care about are logical, so they only take two values — $true$ and $false$ — but we consider a general set of qualitative and continuous relations here

Learning Objective: Known Relations

- Let us suppose that we have **a set P of candidate relations**, where $r \in P$ is defined as either $r : S \rightarrow \mathbb{Z}$ or $r : S \rightarrow \mathbb{R}$, with S a state representation
  - E.g. P can be a set such as $\{leftOf(X,Y), rightOf(X,Y), inFrontOf(X,Y), behind(X,Y)\}$
  - Typically, the relations we care about are logical, so they only take two values — $true$ and $false$ — but we consider a general set of qualitative and continuous relations here
- The learning objective is to **identify a subset $\mathcal{R} \subseteq P$** so that $\mathcal{R}$ is **consistent with a set of training examples**
  - If P only has logical relations, we can interpret $\mathcal{R}$ as $\bigwedge_r r, r \in \mathcal{R}$

Learning Objective: Known Relations

- Let us suppose that we have **a set P of candidate relations**, where $r \in P$ is defined as either $r : S \rightarrow \mathbb{Z}$ or $r : S \rightarrow \mathbb{R}$, with S a state representation
  - E.g. P can be a set such as $\{leftOf(X,Y), rightOf(X,Y), inFrontOf(X,Y), behind(X,Y)\}$
  - Typically, the relations we care about are logical, so they only take two values — $true$ and $false$ — but we consider a general set of qualitative and continuous relations here

- The learning objective is to **identify a subset $\mathcal{R} \subseteq P$** so that $\mathcal{R}$ is **consistent with a set of training examples**
  - If P only has logical relations, we can interpret $\mathcal{R}$ as $\bigwedge_r r, r \in \mathcal{R}$

- Particularly in the context of learning planning domains, we are usually given **a set of plan traces Π**, where each $\pi \in \Pi$ is a sequence of the form $(\mathcal{R}_i, \mathcal{A}_i, \mathcal{E}_i)$, where:
  - $\mathcal{A}_i$ is an action
  - $\mathcal{R}_i$ are the observed preconditions when executing $\mathcal{A}_i$
  - $\mathcal{E}_i$ are the observed effects after executing $\mathcal{A}_i$

Learning Data

- For learning, we consider **ordered vectors** $\boldsymbol{z}_j, 1 \leq j \leq n$, where each element is an instantiated (i.e. ground) relation
  - E.g. considering the previous example set P and the relations defined for objects $X = cup$ and $Y = bottle$, we may have $\boldsymbol{z} = (1, 0, 0, 1)^T$, which we interpret as

$$leftOf(cup, bottle) \wedge \neg rightOf(cup, bottle) \wedge \neg inFrontOf(cup, bottle) \wedge behind(cup, bottle)$$

Learning Data

- For learning, we consider **ordered vectors** $\boldsymbol{z}_j, 1 \leq j \leq n$, where each element is an instantiated (i.e. ground) relation
  - E.g. considering the previous example set P and the relations defined for objects $X = cup$ and $Y = bottle$, we may have $\boldsymbol{z} = (1, 0, 0, 1)^T$, which we interpret as

$$leftOf(cup, bottle) \land \neg rightOf(cup, bottle) \land \neg inFrontOf(cup, bottle) \land behind(cup, bottle)$$

- If all experiences are given as a batch, we consider:
  - **a matrix of** m **examples** $Z \in \mathbb{Z}^{m \times n}$, where each row $Z_i, 1 \leq i \leq m$ is an ordered vector as above
  - **a vector of** m **labels** $\boldsymbol{y}$, where each $\boldsymbol{y}_i \in \{0, 1\}, 1 \leq i \leq m$ denotes whether an experience is positive (1) or negative (0)

Learning Data

- For learning, we consider **ordered vectors** $\boldsymbol{z}_j, 1 \leq j \leq n$, where each element is an instantiated (i.e. ground) relation
  - E.g. considering the previous example set P and the relations defined for objects $X = cup$ and $Y = bottle$, we may have $\boldsymbol{z} = (1, 0, 0, 1)^T$, which we interpret as

$$leftOf(cup, bottle) \wedge \neg rightOf(cup, bottle) \wedge \neg inFrontOf(cup, bottle) \wedge behind(cup, bottle)$$

- If all experiences are given as a batch, we consider:
  - **a matrix of m examples $Z \in \mathbb{Z}^{m \times n}$**, where each row $Z_i, 1 \leq i \leq m$ is an ordered vector as above
  - **a vector of m labels $\boldsymbol{y}$**, where each $\boldsymbol{y}_i \in \{0, 1\}, 1 \leq i \leq m$ denotes whether an experience is positive (1) or negative (0)

- We can, for convenience, split the positive and negative experiences in Z into
  - **Z^+, a matrix of positive examples**, namely examples where $\boldsymbol{y} = 1$
  - **Z^-, a matrix of negative examples**, namely examples where $\boldsymbol{y} = 0$

Learning Data

- For learning, we consider **ordered vectors $\boldsymbol{z}_j, 1 \leq j \leq n$**, where each element is an instantiated (i.e. ground) relation
  - E.g. considering the previous example set P and the relations defined for objects $X = cup$ and $Y = bottle$, we may have $\boldsymbol{z} = (1, 0, 0, 1)^T$, which we interpret as

$$leftOf(cup, bottle) \wedge \neg rightOf(cup, bottle) \wedge \neg inFrontOf(cup, bottle) \wedge behind(cup, bottle)$$

- If all experiences are given as a batch, we consider:
  - **a matrix of m examples $Z \in \mathbb{Z}^{m \times n}$**, where each row $Z_i, 1 \leq i \leq m$ is an ordered vector as above
  - **a vector of m labels $\boldsymbol{y}$**, where each $\boldsymbol{y}_i \in \{0, 1\}, 1 \leq i \leq m$ denotes whether an experience is positive (1) or negative (0)
- We can, for convenience, split the positive and negative experiences in Z into
  - **Z^+, a matrix of positive examples**, namely examples where $\boldsymbol{y} = 1$
  - **Z^-, a matrix of negative examples**, namely examples where $\boldsymbol{y} = 0$
- We will also define $r^{-1}(\boldsymbol{z})$, which is an operator that returns a lifted version of r
  - E.g. $leftOf(cup, bottle)$ is ground, while $leftOf(X, Y)$ is lifted

Overview of Relational Learning Methods

There is a large variety of relational learning methods that can roughly be categorised as follows:

Hypothesis search

Relational learning is observed as a heuristic-driven search problem that looks for rules that describe the plan traces Π, for instance using inductive logic programming

Overview of Relational Learning Methods

There is a large variety of relational learning methods that can roughly be categorised as follows:

Hypothesis search

Relational learning is observed as a heuristic-driven search problem that looks for rules that describe the plan traces Π, for instance using inductive logic programming

Statistical learning

After obtaining a collection of experiences $(Z, \boldsymbol{y})$, a set of relations is learned based on statistical analysis on the observations

Overview of Relational Learning Methods

There is a large variety of relational learning methods that can roughly be categorised as follows:

Hypothesis search

Relational learning is observed as a heuristic-driven search problem that looks for rules that describe the plan traces Π, for instance using inductive logic programming

Online learning

A set of relations is improved continuously, as new experiences $(\boldsymbol{z}, y)$ are collected

Statistical learning

After obtaining a collection of experiences $(Z, \boldsymbol{y})$, a set of relations is learned based on statistical analysis on the observations

Overview of Relational Learning Methods

There is a large variety of relational learning methods that can roughly be categorised as follows:

Hypothesis search

Relational learning is observed as a heuristic-driven search problem that looks for rules that describe the plan traces Π, for instance using inductive logic programming

Statistical learning

After obtaining a collection of experiences $(Z, \boldsymbol{y})$, a set of relations is learned based on statistical analysis on the observations

Online learning

A set of relations is improved continuously, as new experiences $(\boldsymbol{z}, y)$ are collected

Optimisation

Learning is modelled as an optimisation problem over the plan traces Π

Overview of Relational Learning Methods

There is a large variety of relational learning methods that can roughly be categorised as follows:

Hypothesis search

Relational learning is observed as a heuristic-driven search problem that looks for rules that describe the plan traces Π, for instance using inductive logic programming

Statistical learning

After obtaining a collection of experiences $(Z, \boldsymbol{y})$, a set of relations is learned based on statistical analysis on the observations

Online learning

A set of relations is improved continuously, as new experiences $(\boldsymbol{z}, y)$ are collected

Optimisation

Learning is modelled as an optimisation problem over the plan traces Π

Relational reinforcement learning

A variant of reinforcement learning (traditionally of Q-learning), where the state-action value function is learned for relational states and actions

Online Precondition Learning

- We will consider an online learning procedure[1] defined for logical relations that **updates a relation set with each new experience**

[1] X. Wang, "Learning Planning Operators by Observation and Practice," in *Proc. 2nd Int. Conf. Artificial Intelligence Planning Systems*, pp. 1994, 335–341.

Online Precondition Learning

- We will consider an online learning procedure[1] defined for logical relations that **updates a relation set with each new experience**

- The procedure starts with an empty set of relations $\mathcal{R}_0 = \{\}$ and, when the first **positive** execution is observed, **all relations that hold are added to $\mathcal{R}_1$**:

$$\mathcal{R}_1 = \{r^{-1}(\boldsymbol{z}_j) \mid \boldsymbol{z}_j = 1\}, 1 \leq j \leq n$$

  In subsequent experiences, the update depends on the value of y

[1] X. Wang, "Learning Planning Operators by Observation and Practice," in *Proc. 2nd Int. Conf. Artificial Intelligence Planning Systems*, pp. 1994, 335–341.

Online Precondition Learning

- We will consider an online learning procedure[1] defined for logical relations that **updates a relation set with each new experience**

- The procedure starts with an empty set of relations $\mathcal{R}_0 = \{\}$ and, when the first **positive** execution is observed, **all relations that hold are added to $\mathcal{R}_1$**:

$$\mathcal{R}_1 = \{r^{-1}(\boldsymbol{z}_j) \mid \boldsymbol{z}_j = 1\}, 1 \leq j \leq n$$

  In subsequent experiences, the update depends on the value of y

- In the case of a positive experience, **predicates that do not hold are removed from the set**:

$$\mathcal{R}_t = \mathcal{R}_{t-1} \setminus \{r^{-1}(\boldsymbol{z}_j) \mid r^{-1}(\boldsymbol{z}_j) \in \mathcal{R}_{t-1} \wedge \boldsymbol{z}_j = 0\}, 1 \leq j \leq n$$

[1] X. Wang, "Learning Planning Operators by Observation and Practice," in *Proc. 2nd Int. Conf. Artificial Intelligence Planning Systems*, pp. 1994, 335–341.

Online Precondition Learning

- We will consider an online learning procedure[1] defined for logical relations that **updates a relation set with each new experience**

- The procedure starts with an empty set of relations $\mathcal{R}_0 = \{\}$ and, when the first **positive** execution is observed, **all relations that hold are added to $\mathcal{R}_1$**:

$$\mathcal{R}_1 = \{r^{-1}(\boldsymbol{z}_j) \mid \boldsymbol{z}_j = 1\}, 1 \leq j \leq n$$

  In subsequent experiences, the update depends on the value of y

- In the case of a positive experience, **predicates that do not hold are removed from the set**:

$$\mathcal{R}_t = \mathcal{R}_{t-1} \setminus \{r^{-1}(\boldsymbol{z}_j) \mid r^{-1}(\boldsymbol{z}_j) \in \mathcal{R}_{t-1} \wedge \boldsymbol{z}_j = 0\}, 1 \leq j \leq n$$

- In the case of a negative experience in which $\mathcal{R}_{t-1}$ holds before the execution, **the set is expanded by the negations of predicates that hold in z, but which are not in $\mathcal{R}_{t-1}$**:

$$\mathcal{R}_t = \mathcal{R}_{t-1} \cup \{\neg r^{-1}(\boldsymbol{z}_j) \mid r^{-1}(\boldsymbol{z}_j) \notin \mathcal{R}_{t-1} \wedge \boldsymbol{z}_j = 1\}, 1 \leq j \leq n$$

---

[1] X. Wang, "Learning Planning Operators by Observation and Practice," in *Proc. 2nd Int. Conf. Artificial Intelligence Planning Systems*, pp. 1994, 335–341.

Online Effect Learning

- A set of effects $\mathcal{E}$ is learned from what is referred to as a **delta state**, which represents the change in state after an action is executed:

$$\boldsymbol{\Delta} = \boldsymbol{z}_t - \boldsymbol{z}_{t-1}$$

Online Effect Learning

- A set of effects $\mathcal{E}$ is learned from what is referred to as a **delta state**, which represents the change in state after an action is executed:

$$\boldsymbol{\Delta} = \boldsymbol{z}_t - \boldsymbol{z}_{t-1}$$

- As in the case of the preconditions, the effect set is initialised with an empty set $\mathcal{E}_0 = \{\}$, and **all effects that are observed in the delta state are added to $\mathcal{E}$ after the first positive execution**:

$$\mathcal{E}_1 = \{r^{-1}(\boldsymbol{\Delta}_j) \mid \boldsymbol{\Delta}_j \neq 0\}, 1 \leq j \leq n$$

Online Effect Learning

- A set of effects $\mathcal{E}$ is learned from what is referred to as a **delta state**, which represents the change in state after an action is executed:

$$\boldsymbol{\Delta} = \boldsymbol{z}_t - \boldsymbol{z}_{t-1}$$

- As in the case of the preconditions, the effect set is initialised with an empty set $\mathcal{E}_0 = \{\}$, and **all effects that are observed in the delta state are added to $\mathcal{E}$ after the first positive execution**:

$$\mathcal{E}_1 = \{r^{-1}(\boldsymbol{\Delta}_j) \mid \boldsymbol{\Delta}_j \neq 0\}, 1 \leq j \leq n$$

- The effects are only updated in the case of a positive experience, such that **predicates that are part of the delta state are included in $\mathcal{E}$**:

$$\mathcal{E}_t = \mathcal{E}_{t-1} \cup \{r^{-1}(\boldsymbol{\Delta}_j) \mid r^{-1}(\boldsymbol{\Delta}_j) \notin \mathcal{E}_{t-1}\}, 1 \leq j \leq n$$

Online Learning Example

Let us consider a case of learning the preconditions of some action, with
$P = \{leftOf(X, Y), inFrontOf(X, Y), red(X), blue(X)\}$

Online Learning Example

Let us consider a case of learning the preconditions of some action, with
$P = \{leftOf(X, Y), inFrontOf(X, Y), red(X), blue(X)\}$

At $t = 1$, we observe a positive experience:
$\{leftOf(cup, bottle) = true, inFrontOf(cup, bottle) = true, red(cup) = true, blue(bottle) = false\}$
$\implies \boldsymbol{z} = (1, 1, 1, 0)$
$\implies \mathcal{R}_1 = \{leftOf(X, Y), inFrontOf(X, Y), red(X)\}$

Online Learning Example

Let us consider a case of learning the preconditions of some action, with $P = \{leftOf(X, Y), inFrontOf(X, Y), red(X), blue(X)\}$

At $t = 1$, we observe a positive experience:

$$\{leftOf(cup, bottle) = true, inFrontOf(cup, bottle) = true, red(cup) = true, blue(bottle) = false\}$$

$$\implies \boldsymbol{z} = (1, 1, 1, 0)$$

$$\implies \mathcal{R}_1 = \{leftOf(X, Y), inFrontOf(X, Y), red(X)\}$$

At $t = 2$, we observe another positive experience:

$$\{leftOf(cup, bottle) = true, inFrontOf(X, Y) = false, red(cup) = false, blue(bottle) = false\}$$

$$\implies \boldsymbol{z} = (1, 0, 0, 0)$$

$$\implies \mathcal{R}_2 = \{leftOf(X, Y)\}$$

Online Learning Example

Let us consider a case of learning the preconditions of some action, with $P = \{leftOf(X, Y), inFrontOf(X, Y), red(X), blue(X)\}$

At $t = 1$, we observe a positive experience:

$$\{leftOf(cup, bottle) = true, inFrontOf(cup, bottle) = true, red(cup) = true, blue(bottle) = false\}$$

$$\implies \boldsymbol{z} = (1, 1, 1, 0)$$

$$\implies \mathcal{R}_1 = \{leftOf(X, Y), inFrontOf(X, Y), red(X)\}$$

At $t = 2$, we observe another positive experience:

$$\{leftOf(cup, bottle) = true, inFrontOf(X, Y) = false, red(cup) = false, blue(bottle) = false\}$$

$$\implies \boldsymbol{z} = (1, 0, 0, 0)$$

$$\implies \mathcal{R}_2 = \{leftOf(X, Y)\}$$

At $t = 3$, we observe a negative experience:

$$\{leftOf(cup, bottle) = true, inFrontOf(X, Y) = true, red(cup) = true, blue(bottle) = false\}$$

$$\implies \boldsymbol{z} = (1, 1, 1, 0)$$

$$\implies \mathcal{R}_3 = \{leftOf(X, Y), \neg blue(Y)\}$$

Statistical Learning: Qualitative Relations

- In this context, we will consider at a procedure that, given Z^+, looks for relations whose values have **little variation between executions**[2]

[2] N. Abdo et al., "Learning Manipulation Actions From a Few Demonstrations," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2013, pp. 1268–1275.

Statistical Learning: Qualitative Relations

- In this context, we will consider at a procedure that, given Z^+, looks for relations whose values have **little variation between executions**[2]

- For a qualitative relation r_j, the variation between experiences over Z^+ is computed by the **entropy**:
$$J(Z_j^+) = -\sum_v P(Z_j^+ = r_j^v) \log P(Z_j^+ = r_j^v)$$
and r_j is taken to be a relevant precondition or effect if $J(Z_j^+) < \delta$ for a predefined threshold δ

[2] N. Abdo et al., "Learning Manipulation Actions From a Few Demonstrations," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2013, pp. 1268–1275.

Statistical Learning: Qualitative Relations

- In this context, we will consider at a procedure that, given Z^+, looks for relations whose values have **little variation between executions**[2]
- For a qualitative relation r_j, the variation between experiences over Z^+ is computed by the **entropy**:

$$J(Z_j^+) = -\sum_v P(Z_j^+ = r_j^v) \log P(Z_j^+ = r_j^v)$$

  and r_j is taken to be a relevant precondition or effect if $J(Z_j^+) < \delta$ for a predefined threshold δ
- The value r_j^* of r_j that is taken as a precondition or effect is then

$$r_j^* = \arg\max_v P(Z_j^+ = r_j^v)$$

[2] N. Abdo et al., "Learning Manipulation Actions From a Few Demonstrations," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2013, pp. 1268–1275.

Statistical Learning: Continuous Features

- The same procedure[2] is similarly defined for continuous features (e.g. $sizeOf(x)$)

Statistical Learning: Continuous Features

- The same procedure[2] is similarly defined for continuous features (e.g. $sizeOf(x)$)
- Concretely, let us consider a continuous feature f_j whose values are combined in a matrix F_j; the **values in F are then clustered into K clusters** and J is defined as the **average distance to the cluster centroids**:

$$J(F_j^+) = \frac{1}{KN} \sum_{k=1}^{K} \sum_{i=1}^{N_k} d\left(F_{j_i}^{k+}, \boldsymbol{\mu}_k\right)$$

  where $F_{j_i}^{k+}, 1 \leq i \leq N_k$ are points in the positive executions that are assigned to cluster k and d is the Euclidean distance

Statistical Learning: Continuous Features

- The same procedure[2] is similarly defined for continuous features (e.g. $sizeOf(x)$)
- Concretely, let us consider a continuous feature f_j whose values are combined in a matrix F_j; the **values in F are then clustered into K clusters** and J is defined as the **average distance to the cluster centroids**:

$$J(F_j^+) = \frac{1}{KN} \sum_{k=1}^{K} \sum_{i=1}^{N_k} d\left(F_{j_i}^{k+}, \boldsymbol{\mu}_k\right)$$

  where $F_{j_i}^{k+}, 1 \leq i \leq N_k$ are points in the positive executions that are assigned to cluster k and d is the Euclidean distance
- As in the case of qualitative relations, f_j **is taken to be a precondition or effect if** $J(F_j^+) < \delta$

Statistical Learning: Continuous Features

- The same procedure[2] is similarly defined for continuous features (e.g. $sizeOf(x)$)
- Concretely, let us consider a continuous feature f_j whose values are combined in a matrix F_j; the **values in F are then clustered into K clusters** and J is defined as the **average distance to the cluster centroids**:

$$J(F_j^+) = \frac{1}{KN} \sum_{k=1}^{K} \sum_{i=1}^{N_k} d\left(F_{j_i}^{k+}, \boldsymbol{\mu}_k\right)$$

  where $F_{j_i}^{k+}, 1 \leq i \leq N_k$ are points in the positive executions that are assigned to cluster k and d is the Euclidean distance
- As in the case of qualitative relations, **f_j is taken to be a precondition or effect if $J(F_j^+) < \delta$**
- During online use, **the feature vector $\boldsymbol{f}$ is assigned to a cluster**, and **the relation is considered to be satisfied if $d(\boldsymbol{f}, \boldsymbol{\mu}_c) < d_{\max}$**, where $\boldsymbol{\mu}_c$ is the centroid of the cluster and $d_{\max}$ is a predefined distance threshold

Statistical Learning Example: Qualitative Relations

Let us again consider a case of learning the preconditions of some action, with $P = \{leftOf(X, Y), inFrontOf(X, Y), color(X)\}$, with $color \in \{red, green, blue\}$

Statistical Learning Example: Qualitative Relations

Let us again consider a case of learning the preconditions of some action, with $P = \{leftOf(X, Y), inFrontOf(X, Y), color(X)\}$, with $color \in \{red, green, blue\}$

A vector z then encodes the values $(leftOf(X, Y), inFrontOf(X, Y), color(X), color(Y))$, where red takes the value 1, $blue$ has the value 2, and $green$ the value 3

Statistical Learning Example: Qualitative Relations

Let us again consider a case of learning the preconditions of some action, with $P = \{leftOf(X,Y), inFrontOf(X,Y), color(X)\}$, with $color \in \{red, green, blue\}$

A vector z then encodes the values $(leftOf(X,Y), inFrontOf(X,Y), color(X), color(Y))$, where red takes the value 1, $blue$ has the value 2, and $green$ the value 3

Let us assume that we obtain the following observations in positive executions for $X = cup$ and $Y = bottle$:

$$Z^{+T} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 3 & 3 & 2 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 3 & 1 \end{pmatrix}^T$$

Statistical Learning Example: Qualitative Relations

Let us again consider a case of learning the preconditions of some action, with $P = \{leftOf(X,Y), inFrontOf(X,Y), color(X)\}$, with $color \in \{red, green, blue\}$

A vector z then encodes the values $(leftOf(X,Y), inFrontOf(X,Y), color(X), color(Y))$, where red takes the value 1, $blue$ has the value 2, and $green$ the value 3

Let us assume that we obtain the following observations in positive executions for $X = cup$ and $Y = bottle$:

$$Z^{+T} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 3 & 3 & 2 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 3 & 1 \end{pmatrix}^T$$

We then have the following entropy values (with $\ln$):

$leftOf(X,Y)$	$inFrontOf(X,Y)$	$color(X)$	$color(Y)$
$-\infty$	0.6	0.96	0.8

Statistical Learning Example: Qualitative Relations

Let us again consider a case of learning the preconditions of some action, with $P = \{leftOf(X,Y), inFrontOf(X,Y), color(X)\}$, with $color \in \{red, green, blue\}$

A vector z then encodes the values $(leftOf(X,Y), inFrontOf(X,Y), color(X), color(Y))$, where red takes the value 1, $blue$ has the value 2, and $green$ the value 3

Let us assume that we obtain the following observations in positive executions for $X = cup$ and $Y = bottle$:

$$Z^{+T} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 3 & 3 & 2 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 3 & 1 \end{pmatrix}^T$$

We then have the following entropy values (with $\ln$):

$leftOf(X,Y)$	$inFrontOf(X,Y)$	$color(X)$	$color(Y)$
$-\infty$	0.6	0.96	0.8

If we take $\delta = 0.2$, then $\mathcal{R} = leftOf(cup, bottle)$

Learning as Hypothesis Search

- In the hypothesis context search, we will consider a technique[3] that **first learns delta rule predictive models and then looks for general explicit rules that describe the examples in Π**

[3] K. Mourão et al., "Learning STRIPS operators from noisy and incomplete observations," *Proc. 28th Conf. Uncertainty in Artificial Intelligence*, 2012.

Learning as Hypothesis Search

- In the hypothesis context search, we will consider a technique[3] that **first learns delta rule predictive models and then looks for general explicit rules that describe the examples in Π**
- Here, the state is represented by **vectors z, where each $z_i \in \{1, -1, *\}$**, where 1 denotes that a relation is true, -1 that it is false, and $*$ that the value is unknown

[3] K. Mourão et al., "Learning STRIPS operators from noisy and incomplete observations," *Proc. 28th Conf. Uncertainty in Artificial Intelligence*, 2012.

Learning as Hypothesis Search

- In the hypothesis context search, we will consider a technique[3] that **first learns delta rule predictive models and then looks for general explicit rules that describe the examples in Π**
- Here, the state is represented by **vectors $\boldsymbol{z}$, where each $\boldsymbol{z}_i \in \{1, -1, *\}$**, where 1 denotes that a relation is true, -1 that it is false, and $*$ that the value is unknown
- Given Π and vectors $\boldsymbol{z}^{pre}$ and $\boldsymbol{z}^{post}$ that describe the state before and after executing an action, respectively, **a voted perceptron model for each action that predict a delta state**, namely each model is of the form $p_{\mathcal{A}_i}(\boldsymbol{z}) = \Delta$
  - For training the voted perceptron, a binary kernel is used, which has the form $k(\boldsymbol{x}, \boldsymbol{y}) = 2^{\sum_i \mathbb{1}_{x_i = y_i}}$, where $*$ values are not considered

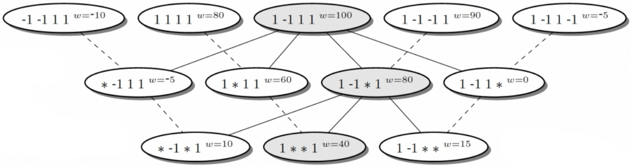
[3] K. Mourão et al., "Learning STRIPS operators from noisy and incomplete observations," *Proc. 28th Conf. Uncertainty in Artificial Intelligence*, 2012.

Learning as Hypothesis Search: Rule Extraction

- Rule search is then performed to extract **explicit action rules** in a two-step process:

Learning as Hypothesis Search: Rule Extraction

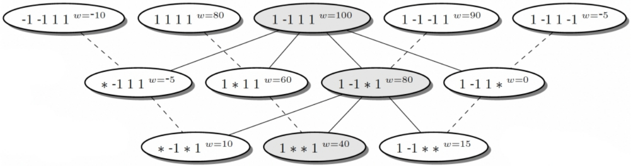
- Rule search is then performed to extract **explicit action rules** in a two-step process:
  - **A precondition set is extracted for each effect**, where the extraction process looks for **the most general rule that predicts the effect** (i.e. a precondition that has as many unknown relation values as possible), where the perceptron weighs the vectors



K. Mourão et al., "Learning STRIPS operators from noisy and incomplete observations," *Proc. 28th Conf. Uncertainty in Artificial Intelligence*, 2012.

Learning as Hypothesis Search: Rule Extraction

- Rule search is then performed to extract **explicit action rules** in a two-step process:
  - **A precondition set is extracted for each effect**, where the extraction process looks for **the most general rule that predicts the effect** (i.e. a precondition that has as many unknown relation values as possible), where the perceptron weighs the vectors



K. Mourão et al., "Learning STRIPS operators from noisy and incomplete observations," *Proc. 28th Conf. Uncertainty in Artificial Intelligence*, 2012.

  - **The rules for the individual effects are merged** in an iterative process that combines preconditions / effects by ensuring that there are no conflicts in the rules, and then attempts to simplify the rules by generalising the relations

Optimisation-Based Learning

Input: a set of plan traces: $T^o = \{t^o\}$.
Output: a domain model m^*.

initialize $\vec{\theta}$ and $\vec{w}$ with random values, and set an iteration number R;
for $r = 1$ to R **do**
  for each $t^o = \langle s_0, a_1^o, \ldots, a_n^o, g \rangle \in T$ **do**
    sample t and m based on $p(t, m|t^o)$;
    calculate the reward $r(t, m)$ based on Equation (7);
    update $\vec{\theta}$ and $\vec{w}$ based on Equations (12) and (13);
  end for
end for
$m^* = \arg\max_{m \in M} w_m$;
return m^*;

An example of an optimisation-based procedure for learning a planning domain. Algorithm from H. H. Zhuo and S. Kambhampati, "Action-model Acquisition from Noisy Plan Traces," in *Proc. 23rd Int. Joint Conf. Artificial Intelligence (IJCAI)*, 2013, pp. 2444–2450.

- Relational learning can also be observed as an optimisation problem that, given a set of plan traces, attempts to **find a domain description by maximising a reward function**

Optimisation-Based Learning

Input: a set of plan traces: $T^o = \{t^o\}$.
Output: a domain model m^*.

```
initialize θ⃗ and w⃗ with random values, and set an iteration number R;
for r = 1 to R do
    for each t^o = ⟨s_0, a_1^o, ..., a_n^o, g⟩ ∈ T do
        sample t and m based on p(t, m|t^o);
        calculate the reward r(t, m) based on Equation (7);
        update θ⃗ and w⃗ based on Equations (12) and (13);
    end for
end for
m* = arg max_{m∈M} w_m;
return m*;
```

An example of an optimisation-based procedure for learning a planning domain. Algorithm from H. H. Zhuo and S. Kambhampati, "Action-model Acquisition from Noisy Plan Traces," in *Proc. 23rd Int. Joint Conf. Artificial Intelligence (IJCAI)*, 2013, pp. 2444–2450.

- Relational learning can also be observed as an optimisation problem that, given a set of plan traces, attempts to **find a domain description by maximising a reward function**
- This can be done by **defining a parameterised distribution over models** and an associated parameterised optimisation objective, for instance evaluating successful executions under a given model

Optimisation-Based Learning

Input: a set of plan traces: $T^o = \{t^o\}$.
Output: a domain model m^*.

initialize $\vec{\theta}$ and $\vec{w}$ with random values, and set an iteration number R;
for $r = 1$ to R **do**
  for each $t^o = \langle s_0, a_1^o, \ldots, a_n^o, g \rangle \in T$ **do**
    sample t and m based on $p(t, m|t^o)$;
    calculate the reward $r(t, m)$ based on Equation (7);
    update $\vec{\theta}$ and $\vec{w}$ based on Equations (12) and (13);
  end for
end for
$m^* = \arg\max_{m \in M} w_m$;
return m^*;

An example of an optimisation-based procedure for learning a planning domain. Algorithm from H. H. Zhuo and S. Kambhampati, "Action-model Acquisition from Noisy Plan Traces," in *Proc. 23rd Int. Joint Conf. Artificial Intelligence (IJCAI)*, 2013, pp. 2444–2450.

- Relational learning can also be observed as an optimisation problem that, given a set of plan traces, attempts to **find a domain description by maximising a reward function**
- This can be done by **defining a parameterised distribution over models** and an associated parameterised optimisation objective, for instance evaluating successful executions under a given model
- If a gradient-based optimisation procedure is used, this type of relational learning can also be performed in the context of neural networks

Relational Reinforcement Learning

Initialize $\hat{Q}_0$ to assign 0 to all (s, a) pairs
Initialize Examples to the empty set.
e := 0
do forever
  e := e + 1
  i := 0
  generate a random state s_0
  while not goal(s_i) **do**
    select an action a_i stochastically
      using the Q-exploration strategy from Equation (1)
      using the current hypothesis for $\hat{Q}_e$
    perform action a_i
    receive an immediate reward $r_i = r(s_i, a_i)$
    observe the new state s_{i+1}
    i:=i+1
  endwhile
  for j=i−1 to 0 **do**
    generate example $x = (s_j, a_j, \hat{q}_j)$,
      where $\hat{q}_j := r_j + \gamma max_{a'} \hat{Q}_e(s_{j+1}, a')$
    if an example $(s_j, a_j, \hat{q}_{old})$ exists in Examples, replace it with x,
      else add x to Examples
  update $\hat{Q}_e$ using TILDE-RT to produce $\hat{Q}_{e+1}$ using Examples

S. Dzeroski, L. De Raedt, and H. Blockeel, "Relational reinforcement learning," *Proc. 8th Int. Conf. Inductive Logic Programming (ILP)*, 1998.

- In relational reinforcement learning, the objective is similar as in typical reinforcement learning, namely RRL aims to **identify a policy $\pi : S \rightarrow A$ that maximises an expected reward**

Relational Reinforcement Learning

```
Initialize Q̂_0 to assign 0 to all (s, a) pairs
Initialize Examples to the empty set.
e := 0
do forever
  e := e + 1
  i := 0
  generate a random state s_0
  while not goal(s_i) do
    select an action a_i stochastically
      using the Q-exploration strategy from Equation (1)
      using the current hypothesis for Q̂_e
    perform action a_i
    receive an immediate reward r_i = r(s_i, a_i)
    observe the new state s_{i+1}
    i:=i+1
  endwhile
  for j=i−1 to 0 do
    generate example x = (s_j, a_j, q̂_j),
      where q̂_j := r_j + γmax_{a'} Q̂_e(s_{j+1}, a')
    if an example (s_j, a_j, q̂_old) exists in Examples, replace it with x,
      else add x to Examples
  update Q̂_e using TILDE-RT to produce Q̂_{e+1} using Examples
```

S. Dzeroski, L. De Raedt, and H. Blockeel, "Relational reinforcement learning," *Proc. 8th Int. Conf. Inductive Logic Programming (ILP)*, 1998.

- In relational reinforcement learning, the objective is similar as in typical reinforcement learning, namely RRL aims to **identify a policy $\pi : S \rightarrow A$ that maximises an expected reward**
- The difference with traditional reinforcement learning is in the Q-function representation — in RRL, **a logical decision tree is used to model the function**, where each node in the tree is **a binary decision node**

Relational Reinforcement Learning

```
Initialize Q̂_0 to assign 0 to all (s, a) pairs
Initialize Examples to the empty set.
e := 0
do forever
  e := e + 1
  i := 0
  generate a random state s_0
  while not goal(s_i) do
    select an action a_i stochastically
      using the Q-exploration strategy from Equation (1)
      using the current hypothesis for Q̂_e
    perform action a_i
    receive an immediate reward r_i = r(s_i, a_i)
    observe the new state s_{i+1}
    i:=i+1
  endwhile
  for j=i−1 to 0 do
    generate example x = (s_j, a_j, q̂_j),
      where q̂_j := r_j + γmax_{a'} Q̂_e(s_{j+1}, a')
    if an example (s_j, a_j, q̂_old) exists in Examples, replace it with x,
      else add x to Examples
  update Q̂_e using TILDE-RT to produce Q̂_{e+1} using Examples
```

S. Dzeroski, L. De Raedt, and H. Blockeel, "Relational reinforcement learning," *Proc. 8th Int. Conf. Inductive Logic Programming (ILP)*, 1998.

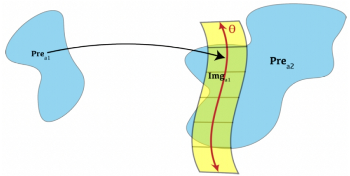
- In relational reinforcement learning, the objective is similar as in typical reinforcement learning, namely RRL aims to **identify a policy $\pi : S \rightarrow A$ that maximises an expected reward**
- The difference with traditional reinforcement learning is in the Q-function representation — in RRL, **a logical decision tree is used to model the function**, where each node in the tree is **a binary decision node**
- As is typical in reinforcement learning, the Q-function is continuously updated; in RRL, the update is performed **after finding a sequence of actions that lead to a desired goal**
  - **The tree is induced from a set of examples that is regularly updated** (much like an experience buffer)

Symbol Learning

- For all previous approaches, we made **an assumption that the set of relations P from which we extract relational rules is given**, but this puts a manual design burden
  - The manual definition of relations is challenging particularly because we cannot always guarantee that the used set contains all **relevant relations** for describing a problem

Symbol Learning

- For all previous approaches, we made **an assumption that the set of relations P from which we extract relational rules is given**, but this puts a manual design burden
  - The manual definition of relations is challenging particularly because we cannot always guarantee that the used set contains all **relevant relations** for describing a problem

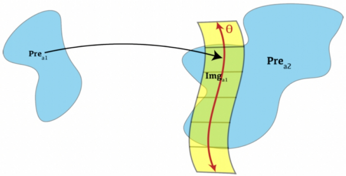


B. Ames, A. Thackston and G. Konidaris, "Learning Symbolic Representations for Planning with Parameterized Skills," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018, pp. 526–533.

- In **symbol learning**, the objective is that of automatically identifying relations that can be embedded in skill models — learning thus has to be performed in an unsupervised or self-supervised manner

Symbol Learning

- For all previous approaches, we made **an assumption that the set of relations P from which we extract relational rules is given**, but this puts a manual design burden
  - The manual definition of relations is challenging particularly because we cannot always guarantee that the used set contains all **relevant relations** for describing a problem



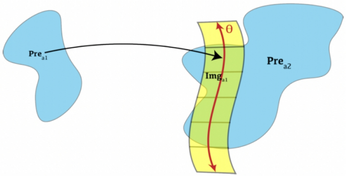
B. Ames, A. Thackston and G. Konidaris, "Learning Symbolic Representations for Planning with Parameterized Skills," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018, pp. 526–533.

- In **symbol learning**, the objective is that of automatically identifying relations that can be embedded in skill models — learning thus has to be performed in an unsupervised or self-supervised manner

- There is no general solution to the symbol learning problem — the problem is too unconstrained — but there are existing approaches for it in the context of **parameterised skills**

Symbol Learning

- For all previous approaches, we made **an assumption that the set of relations P from which we extract relational rules is given**, but this puts a manual design burden
  - The manual definition of relations is challenging particularly because we cannot always guarantee that the used set contains all **relevant relations** for describing a problem



B. Ames, A. Thackston and G. Konidaris, "Learning Symbolic Representations for Planning with Parameterized Skills," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018, pp. 526–533.

- In **symbol learning**, the objective is that of automatically identifying relations that can be embedded in skill models — learning thus has to be performed in an unsupervised or self-supervised manner
- There is no general solution to the symbol learning problem — the problem is too unconstrained — but there are existing approaches for it in the context of **parameterised skills**
- In this context, **symbols are learned in terms of the skill parameters**, for instance using clustering, and can be seen as **discretisations of the parameter space**
  - The learned symbols enable transitions between the created clusters

Summary

- Relational learning is a problem of extracting relational rules from observations, typically plan traces
- Relational learning can be used to learn skill models (namely skill initiation and termination conditions), but also to learn execution rules
- There are various relational learning methods, which we categorised into hypothesis search, online learning, statistical learning, optimisation-based, and relational reinforcement learning
- Symbol learning is the problem of acquiring models of relations, thereby eliminating the need for manual specification of relations