

# General Acyclicity and Cyclicity Notions for the Disjunctive Skolem Chase (Extended Abstract)

**Lukas Gerlach<sup>1</sup>**

**David Carral<sup>2</sup>**

<sup>1</sup>Knowledge-Based Systems, TU Dresden

<sup>2</sup>LIRMM, Inria, University of Montpellier, CNRS

02.09.2023



International Center  
for Computational Logic

*Inria*

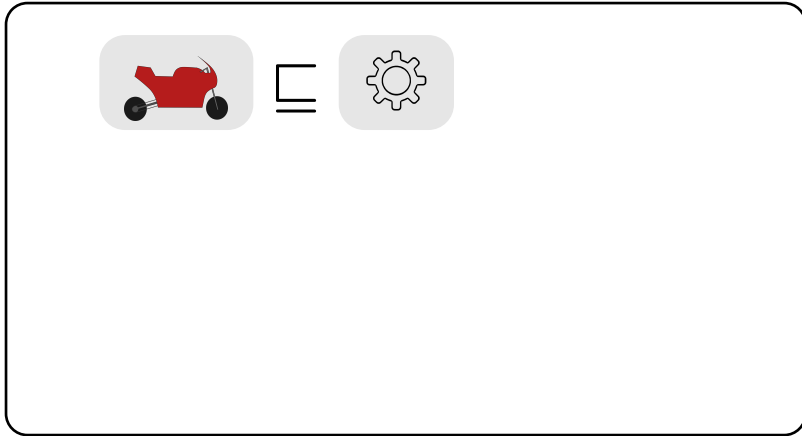
# Running Riding Example

---

*Structurally inspired by 00007.owl from OXFD:*

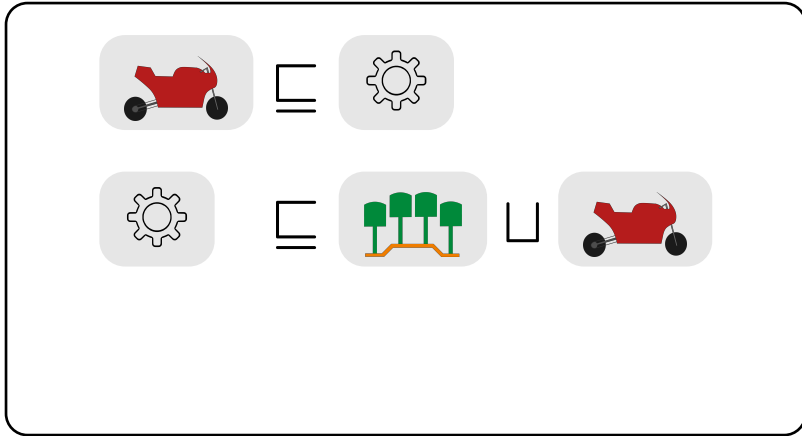
# Running Riding Example

*Structurally inspired by 00007.owl from OXFD:*



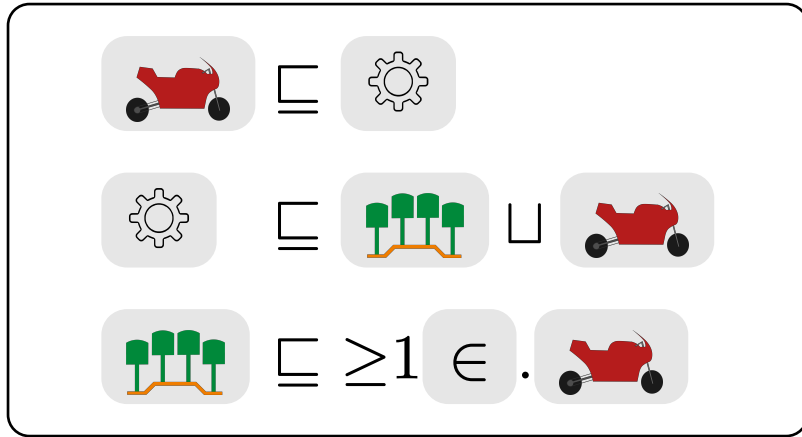
# Running Riding Example

*Structurally inspired by 00007.owl from OXFD:*



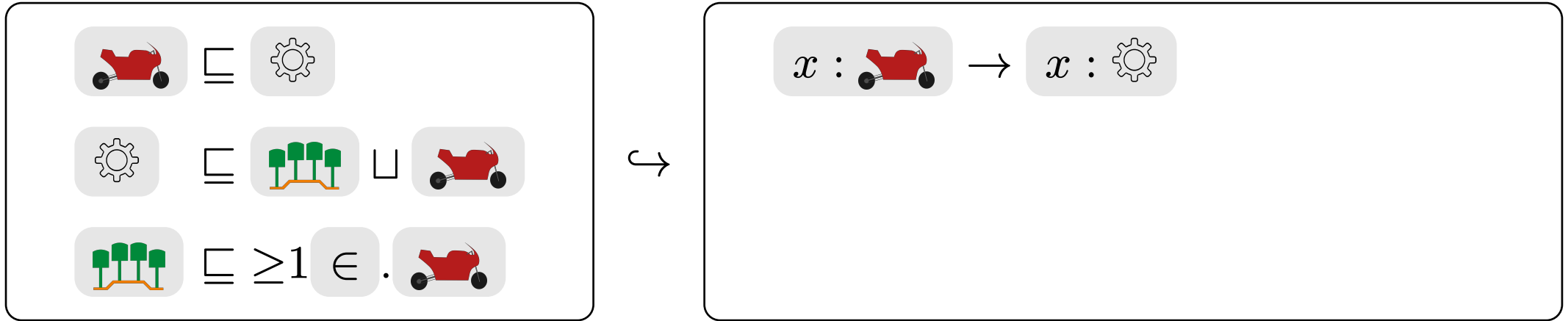
# Running Riding Example

Structurally inspired by *00007.owl* from *OXFD*:



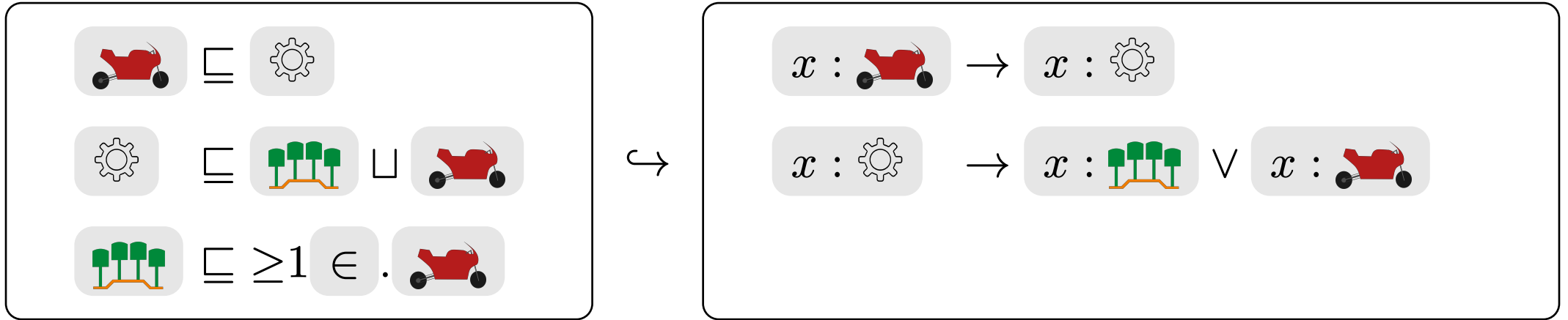
# Running Riding Example

Structurally inspired by *00007.owl* from *OXFD*:



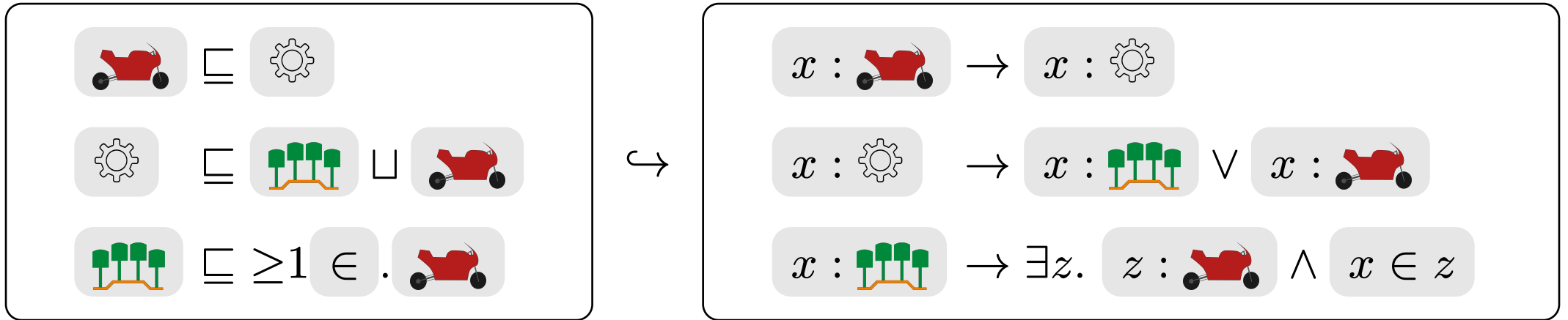
# Running Riding Example

Structurally inspired by *00007.owl* from *OXFD*:



# Running Riding Example

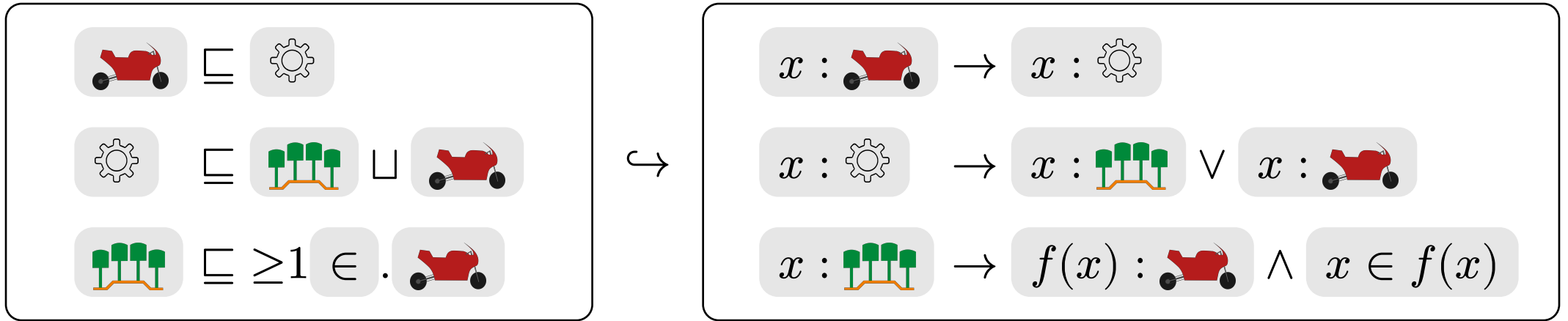
Structurally inspired by *00007.owl* from *OXFD*:





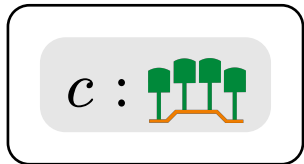
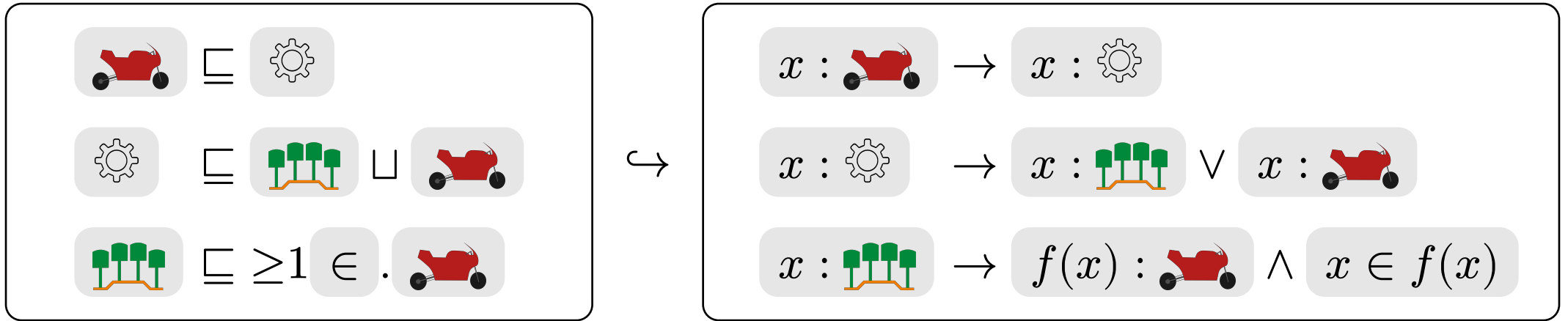
# Running Riding Example

Structurally inspired by *00007.owl* from *OXFD*:



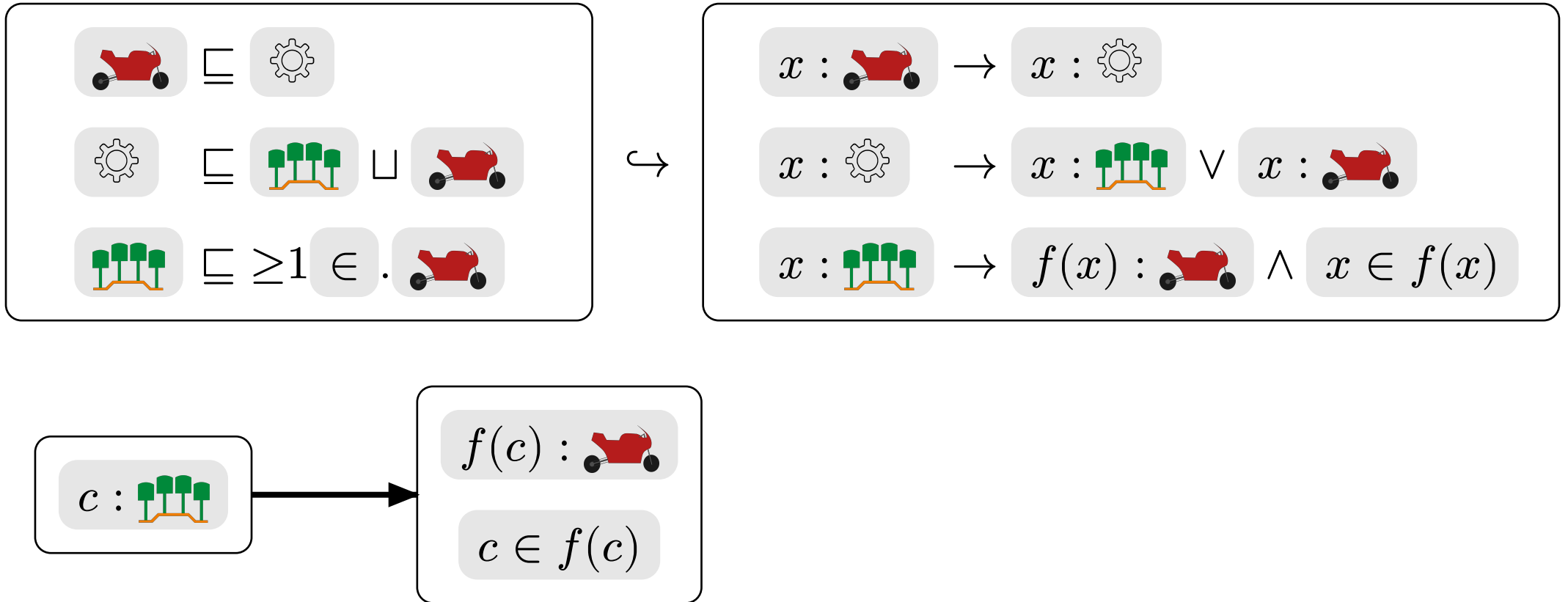
# Running Riding Example

Structurally inspired by *00007.owl* from *OXFD*:



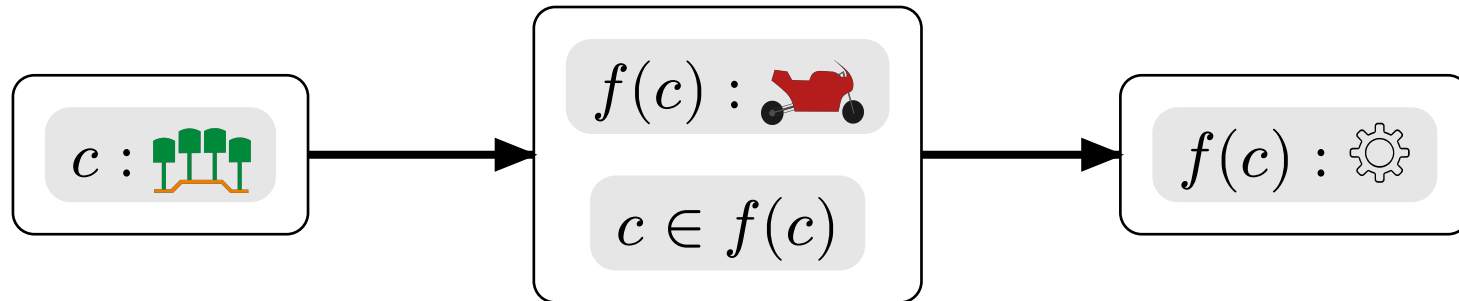
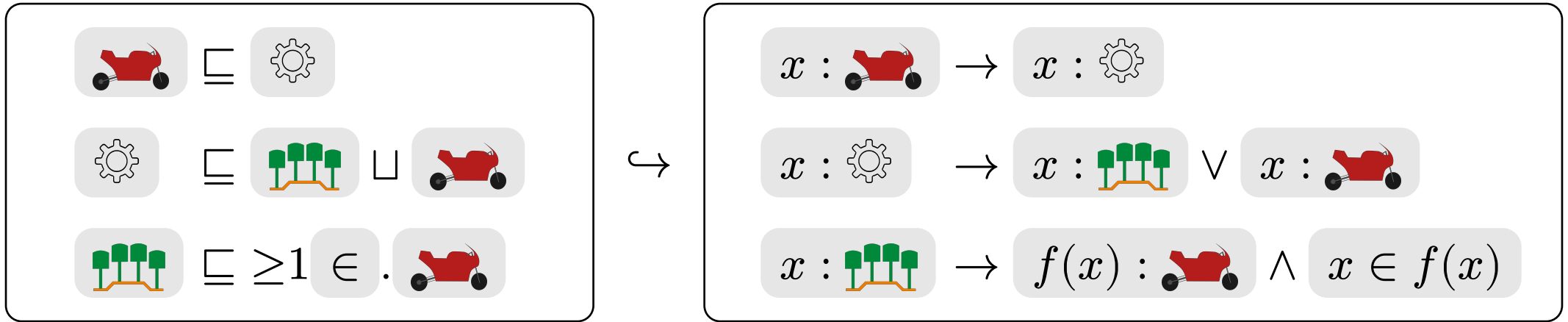
# Running Riding Example

Structurally inspired by *00007.owl* from *OXFD*:



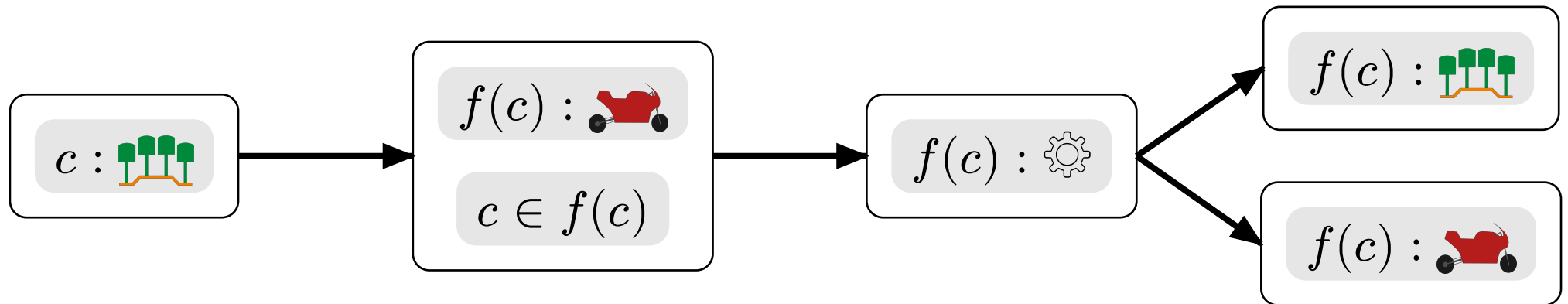
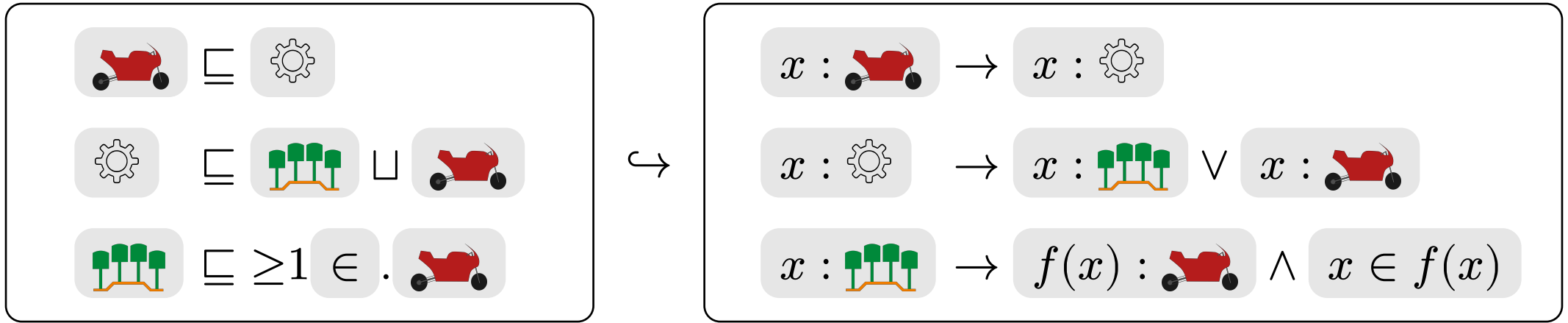
# Running Riding Example

Structurally inspired by *00007.owl* from *OXFD*:



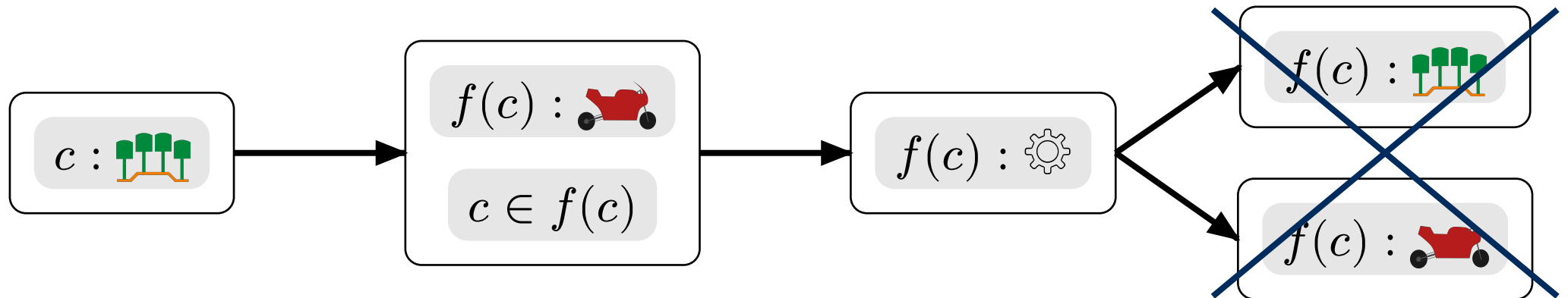
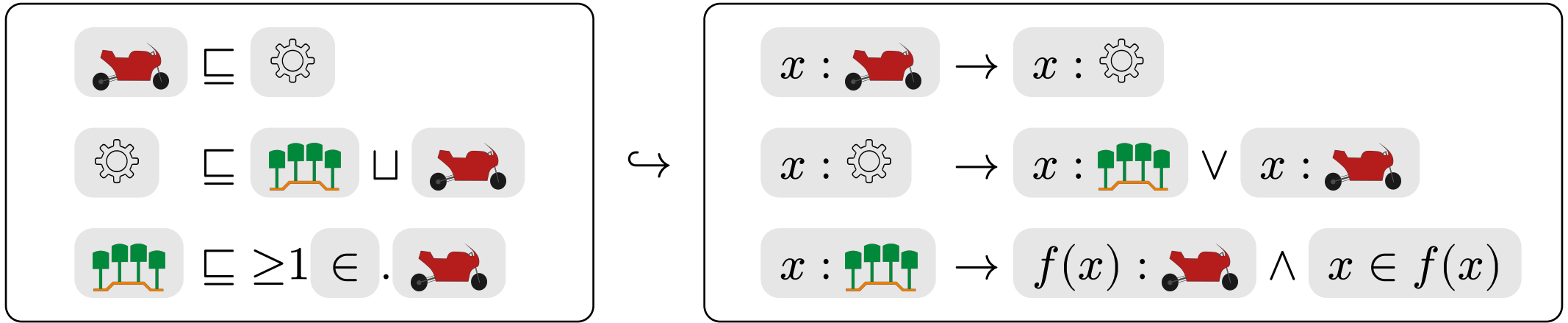
# Running Riding Example

Structurally inspired by *00007.owl* from *OXFD*:

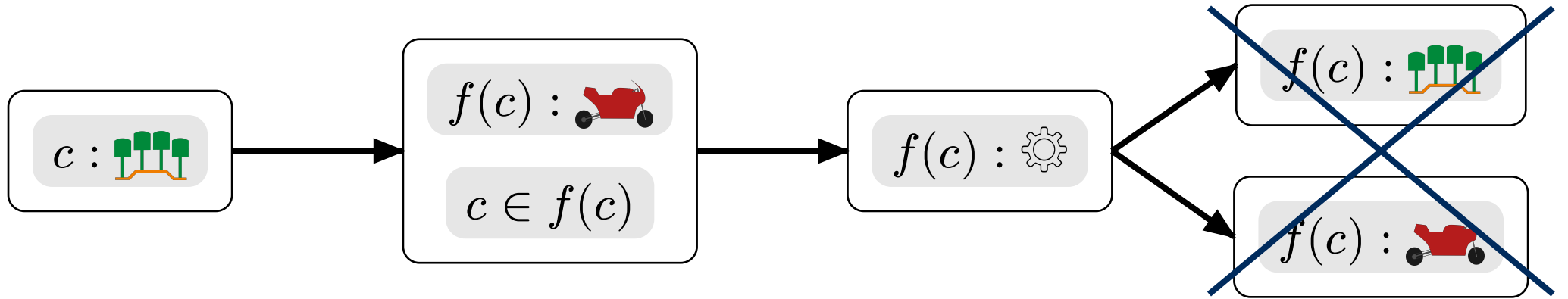


# Running Riding Example

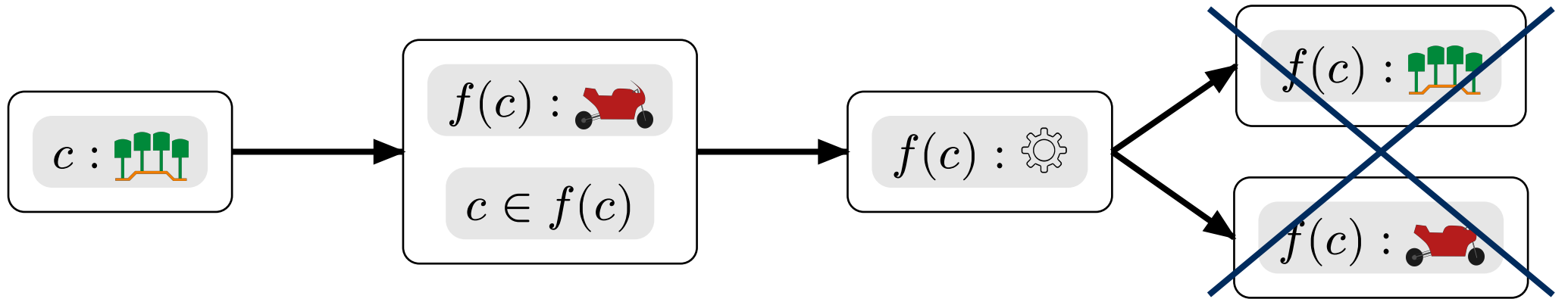
Structurally inspired by *00007.owl* from *OXFD*:



# About that Chase...



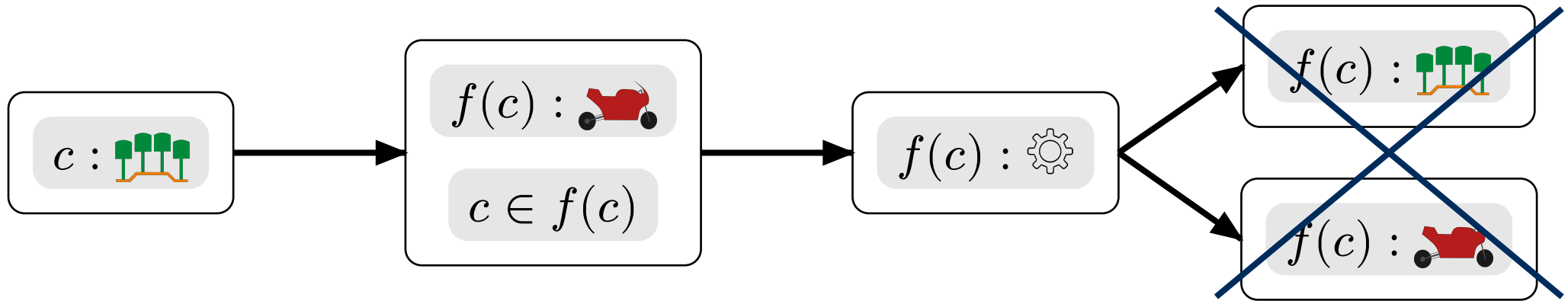
# About that Chase...



😊 Leaves of tree form *universal model set* (can answer queries) [Bou+16]

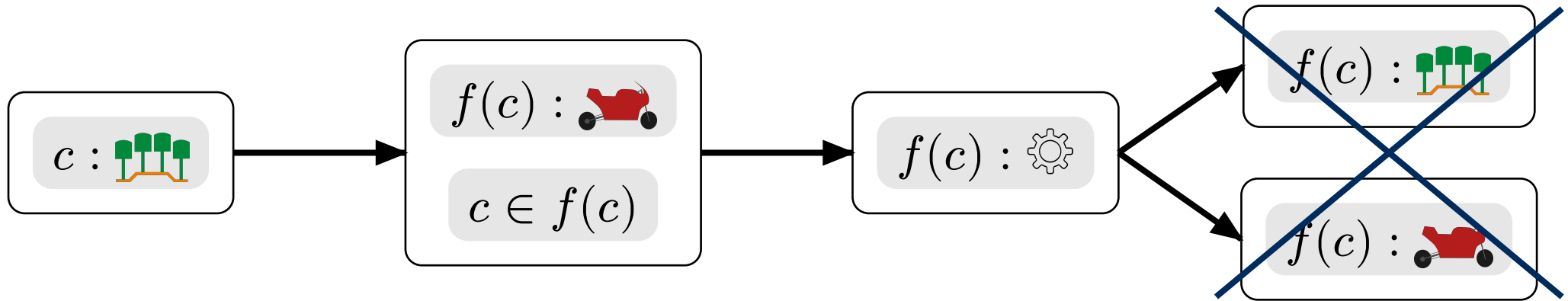


# About that Chase...



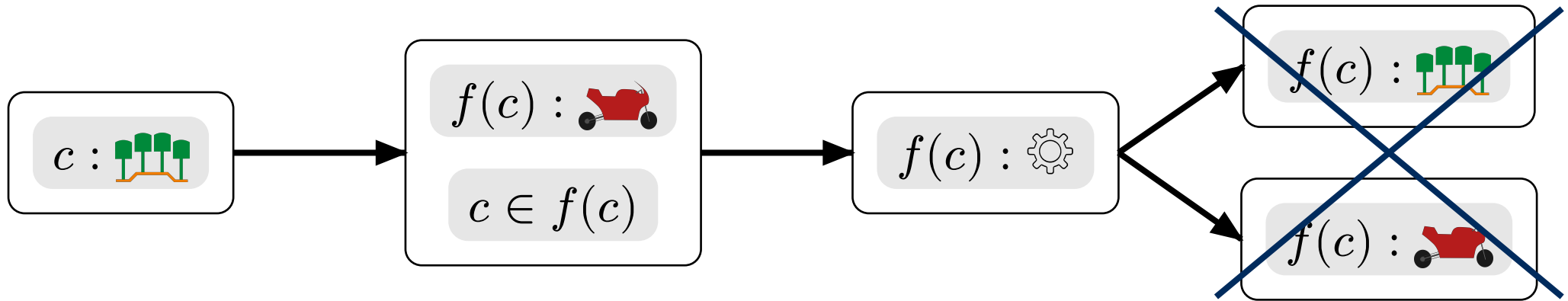
- 😊 Leaves of tree form *universal model set* (can answer queries) [Bou+16]
- 🤔 But query answering/entailment is undecidable [BV81]

# About that Chase...



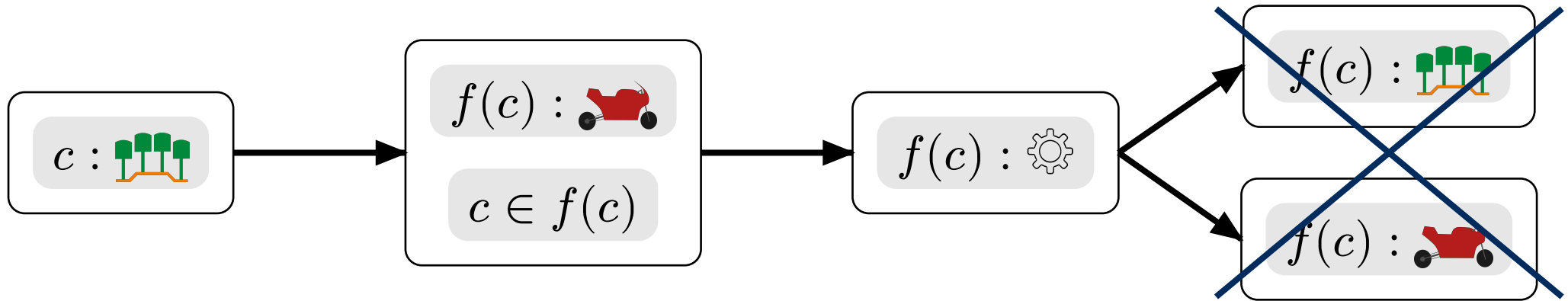
- 😊 Leaves of tree form *universal model set* (can answer queries) [Bou+16]
- 🤔 But query answering/entailment is undecidable [BV81]
- 😞 Chase termination is also undecidable [GM14, G018]

# About that Chase...



- 😊 Leaves of tree form *universal model set* (can answer queries) [Bou+16]
- 🤔 But query answering/entailment is undecidable [BV81]
- 😬 Chase termination is also undecidable [GM14, G018]
- 😬 MFA/MFC can sometimes detect (non-)termination [Cue+13, CDK17]

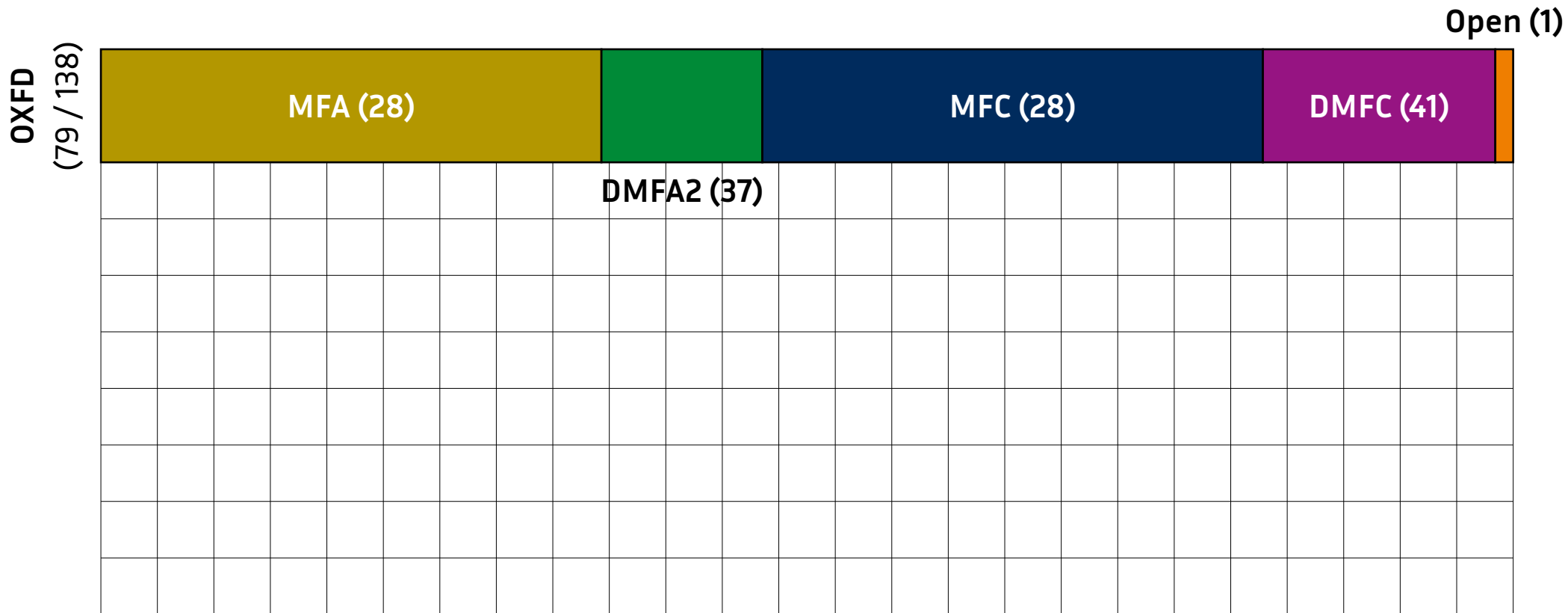
# About that Chase...



- 😊 Leaves of tree form *universal model set* (can answer queries) [Bou+16]
- 🤔 But query answering/entailment is undecidable [BV81]
- 😬 Chase termination is also undecidable [GM14, G018]
- 😬 MFA/MFC can sometimes detect (non-)termination [Cue+13, CDK17]
- 🧐 We use ideas from RMFA/RMFC (for a different chase variant) [CDK17] to improve MFA/MFC for disjunctions: DMFA(2)/DMFC [GC23a]

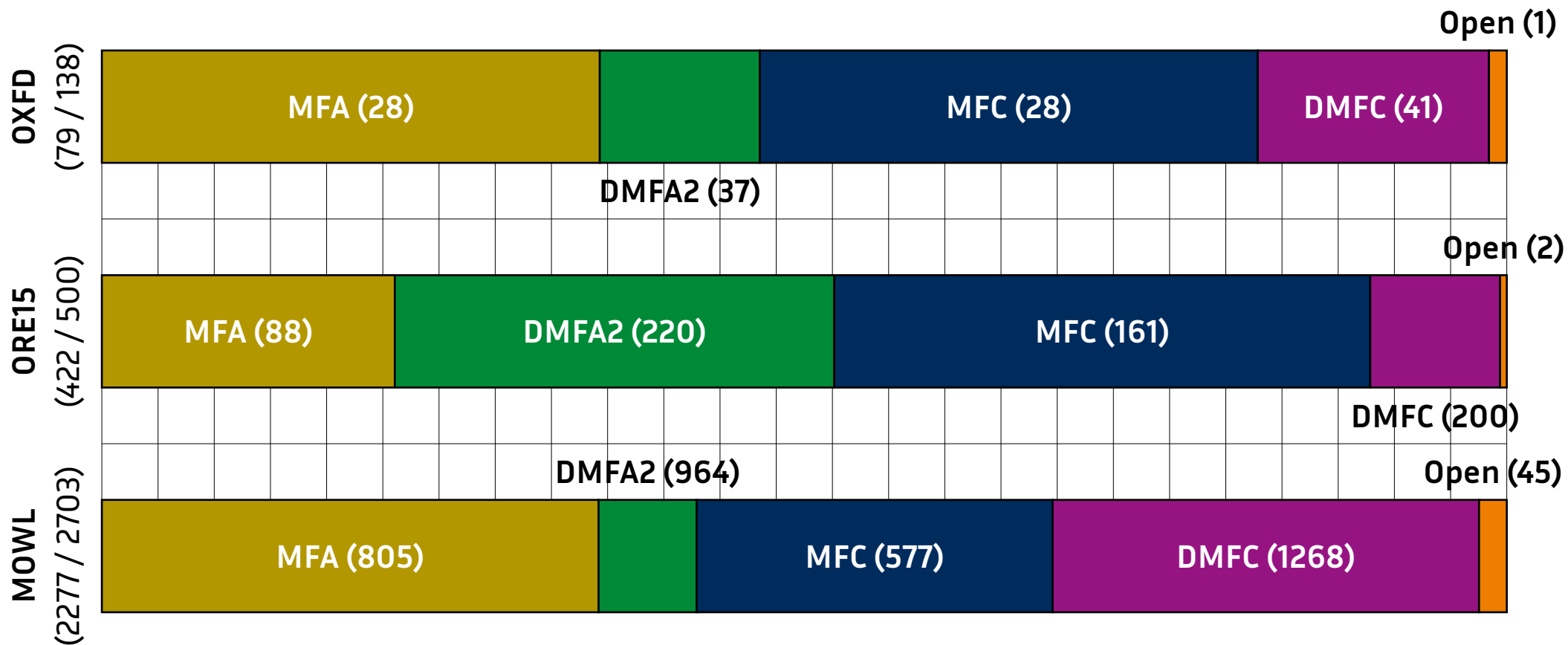
# Is it any good?

```
cat my-ruleset | docker run --rm -i registry.gitlab.com/m0nstr/dmfa-checker -t [non_]termination -cv skolem [-disj] [-depth 2]
```



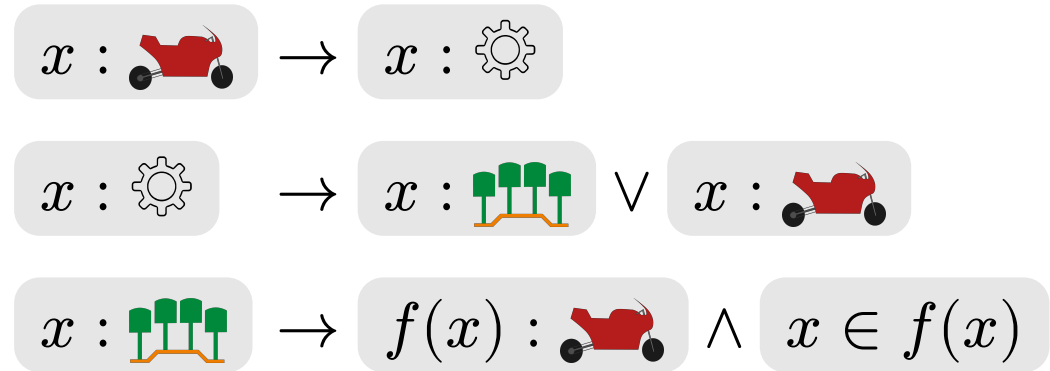
# Is it any good?

```
cat my-ruleset | docker run --rm -i registry.gitlab.com/m0nstr/dmfa-checker -t [non_]termination -cv skolem [-disj] [-depth 2]
```



# How does Model Faithful Acyclicity work?

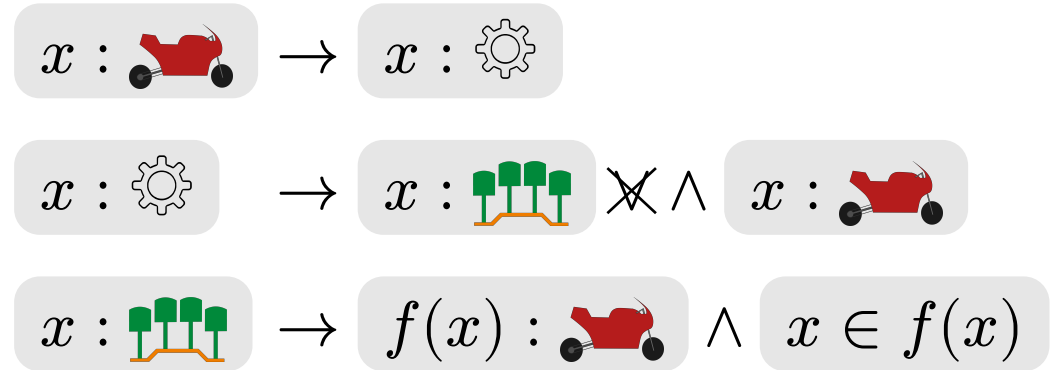
*We want to know if the chase terminates on all databases for a rule set.*



# How does Model Faithful Acyclicity work?

*We want to know if the chase terminates on all databases for a rule set.*

## 1. Replace Disjunctions

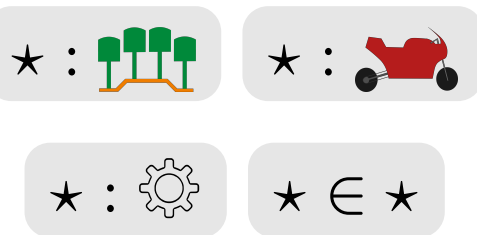
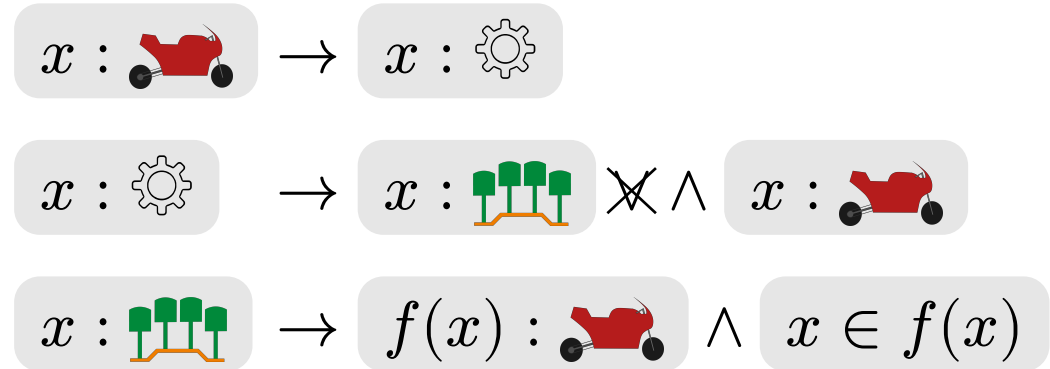




# How does Model Faithful Acyclicity work?

*We want to know if the chase terminates on all databases for a rule set.*

1. Replace Disjunctions
2. Use Critical Database



# How does Model Faithful Acyclicity work?

We want to know if the chase terminates on all databases for a rule set.

1. Replace Disjunctions
2. Use Critical Database
3. Run Chase

$$x : \text{🏍️} \rightarrow x : \text{⚙️}$$

$$x : \text{⚙️} \rightarrow x : \text{🌳} \not\wedge x : \text{🏍️}$$

$$x : \text{🌳} \rightarrow f(x) : \text{🏍️} \wedge x \in f(x)$$

$$\star : \text{🌳} \quad \star : \text{🏍️}$$

$$\star : \text{⚙️} \quad \star \in \star$$

$$f(\star) : \text{🏍️} \quad \star \in f(\star)$$

# How does Model Faithful Acyclicity work?

We want to know if the chase terminates on all databases for a rule set.

1. Replace Disjunctions
2. Use Critical Database
3. Run Chase

$$x : \text{motorcycle} \rightarrow x : \text{gear}$$

$$x : \text{gear} \rightarrow x : \text{forest} \not\wedge x : \text{motorcycle}$$

$$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$

$$\star : \text{forest} \quad \star : \text{motorcycle}$$

$$\star : \text{gear} \quad \star \in \star$$

$$f(\star) : \text{motorcycle} \quad \star \in f(\star)$$

$$f(\star) : \text{gear}$$

# How does Model Faithful Acyclicity work?

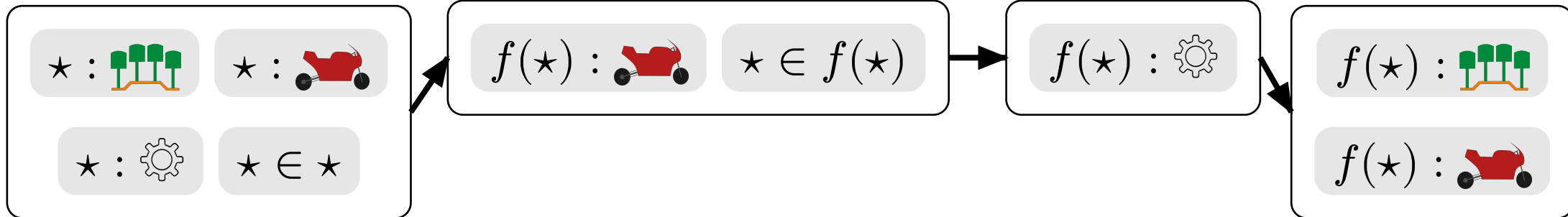
We want to know if the chase terminates on all databases for a rule set.

1. Replace Disjunctions
2. Use Critical Database
3. Run Chase

$$x : \text{motorcycle} \rightarrow x : \text{gear}$$

$$x : \text{gear} \rightarrow x : \text{forest} \not\wedge x : \text{motorcycle}$$

$$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$



# How does Model Faithful Acyclicity work?

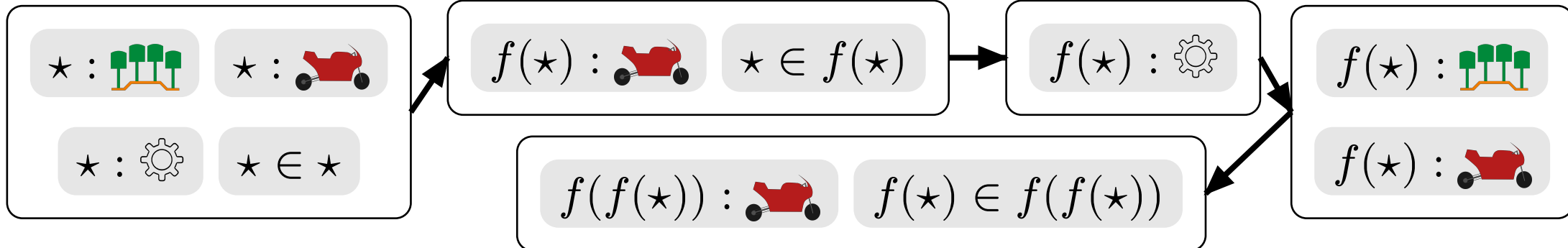
We want to know if the chase terminates on all databases for a rule set.

1. Replace Disjunctions
2. Use Critical Database
3. Run Chase

$$x : \text{🏍️} \rightarrow x : \text{⚙️}$$

$$x : \text{⚙️} \rightarrow x : \text{🌳} \not\wedge x : \text{🏍️}$$

$$x : \text{🌳} \rightarrow f(x) : \text{🏍️} \wedge x \in f(x)$$



# How does Model Faithful Acyclicity work?

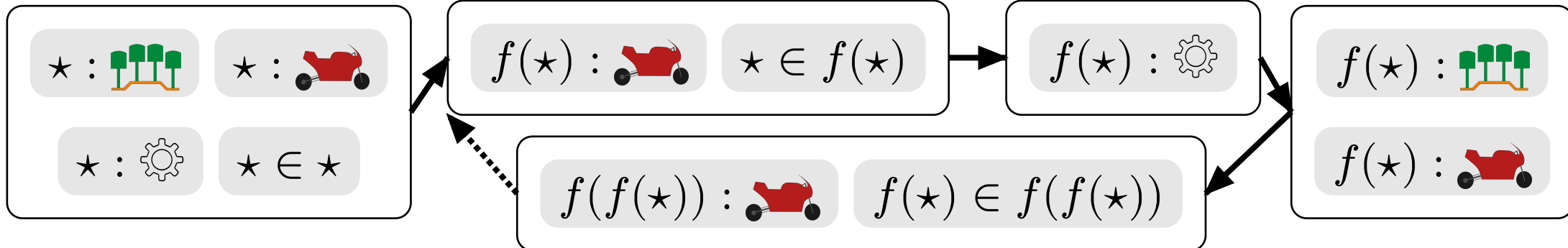
We want to know if the chase terminates on all databases for a rule set.

1. Replace Disjunctions
2. Use Critical Database
3. Run Chase
4. Stop on Cyclic Term

$$x : \text{motorcycle} \rightarrow x : \text{gear}$$

$$x : \text{gear} \rightarrow x : \text{forest} \not\wedge x : \text{motorcycle}$$

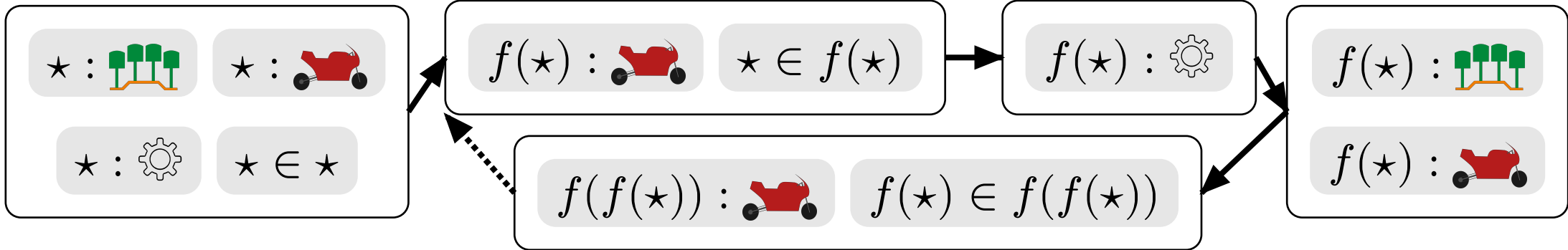
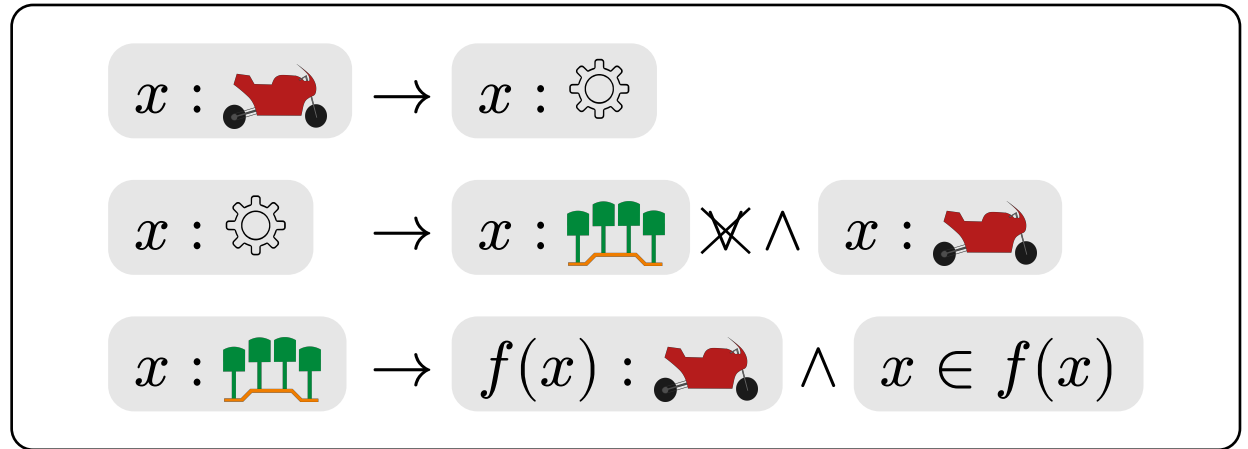
$$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$



# How does Model Faithful Acyclicity work?

We want to know if the chase terminates on all databases for a rule set.

1. Replace Disjunctions
2. Use Critical Database
3. Run Chase
4. Stop on Cyclic Term
5. MFA iff no Cyclic Term

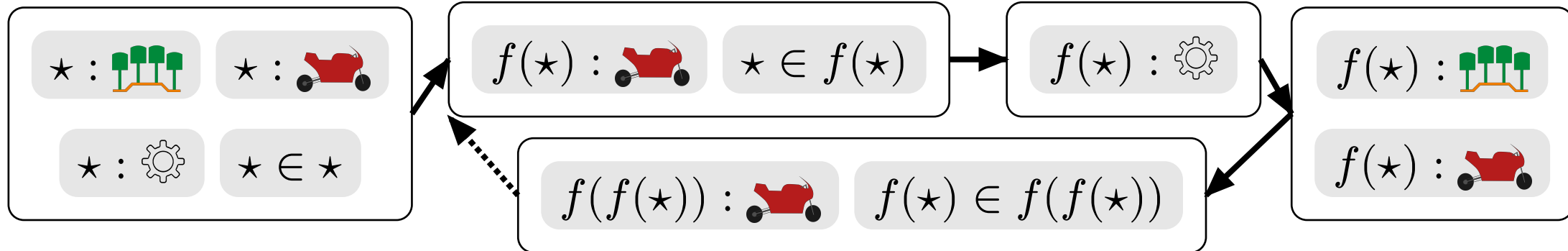


# How does Model Faithful Acyclicity work?

*We want to know if the chase terminates on all databases for a rule set.*

1. Replace Disjunctions
2. Use Critical Database
3. Run Chase
4. Stop on Cyclic Term
5. MFA iff no Cyclic Term

**Theorem.** If a rule set  $R$  is MFA, then  $R$  terminates on every database.  
*Intuitively, the critical database chase subsumes every possible chase tree.*





# How does Disjunctive MFA work?

*We want to know if the chase terminates on all databases for a rule set.*

$$x : \text{moped} \rightarrow x : \text{gear}$$

$$x : \text{gear} \rightarrow x : \text{forest} \vee x : \text{moped}$$

$$x : \text{forest} \rightarrow f(x) : \text{moped} \wedge x \in f(x)$$

# How does Disjunctive MFA work?

*We want to know if the chase terminates on all databases for a rule set.*

## 1. ~~Replace Disjunctions~~

$$x : \text{🏍️} \rightarrow x : \text{⚙️}$$

$$x : \text{⚙️} \rightarrow x : \text{🌳🌳🌳} \vee x : \text{🏍️}$$

$$x : \text{🌳🌳🌳} \rightarrow f(x) : \text{🏍️} \wedge x \in f(x)$$

# How does Disjunctive MFA work?

We want to know if the chase terminates on all databases for a rule set.

## 1. Use Critical Database

$$x : \text{motorcycle} \rightarrow x : \text{gear}$$

$$x : \text{gear} \rightarrow x : \text{forest} \vee x : \text{motorcycle}$$

$$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$



# How does Disjunctive MFA work?

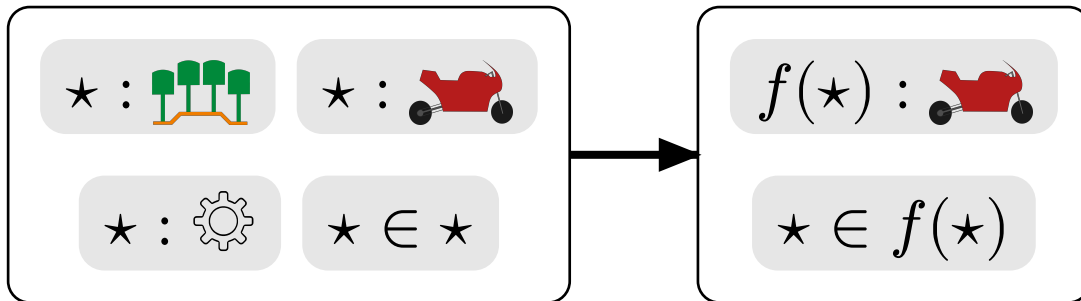
We want to know if the chase terminates on all databases for a rule set.

1. Use Critical Database
2. Run Chase  
**without blocked triggers**

$$x : \text{motorcycle} \rightarrow x : \text{gear}$$

$$x : \text{gear} \rightarrow x : \text{forest} \vee x : \text{motorcycle}$$

$$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$



# How does Disjunctive MFA work?

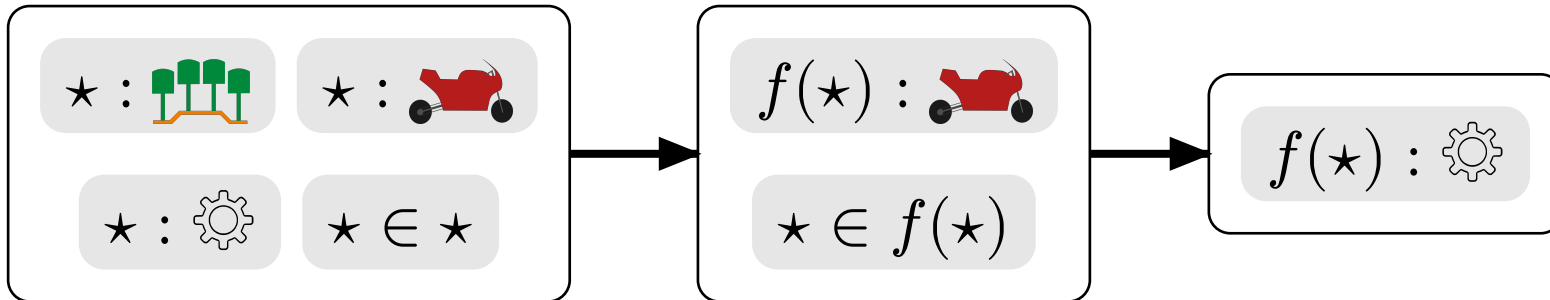
We want to know if the chase terminates on all databases for a rule set.

1. Use Critical Database
2. Run Chase  
**without blocked triggers**

$$x : \text{motorcycle} \rightarrow x : \text{gear}$$

$$x : \text{gear} \rightarrow x : \text{forest} \vee x : \text{motorcycle}$$

$$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$



# How does Disjunctive MFA work?

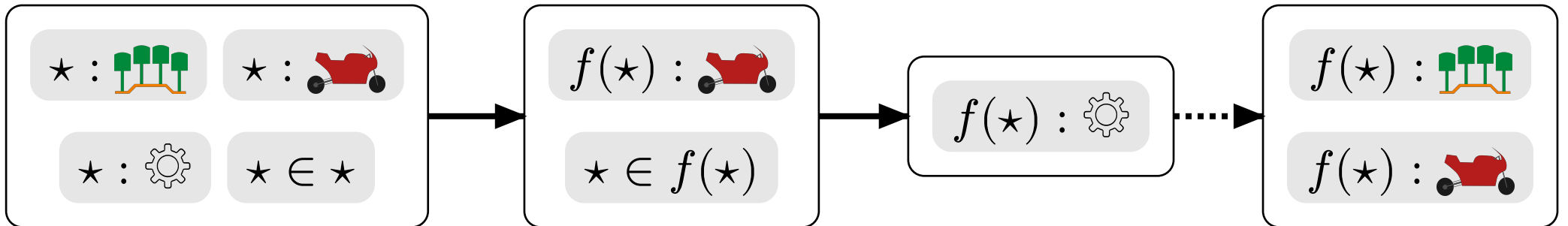
We want to know if the chase terminates on all databases for a rule set.

1. Use Critical Database
2. Run Chase  
**without blocked triggers**

$$x : \text{motorcycle} \rightarrow x : \text{gear}$$

$$x : \text{gear} \rightarrow x : \text{forest} \vee x : \text{motorcycle}$$

$$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$



# How does Disjunctive MFA work?

*We want to know if the chase terminates on all databases for a rule set.*

1. Use Critical Database
2. Run Chase  
**without blocked triggers**

$$\begin{aligned}x : \text{🏍️} &\rightarrow x : \text{⚙️} \\x : \text{⚙️} &\rightarrow x : \text{🌳🌳🌳} \vee x : \text{🏍️} \\x : \text{🌳🌳🌳} &\rightarrow f(x) : \text{🏍️} \wedge x \in f(x)\end{aligned}$$

Is  $x : \text{⚙️} \rightarrow x : \text{🌳🌳🌳} \vee x : \text{🏍️}$  with  $x/f(\star)$  blocked?

# How does Disjunctive MFA work?

We want to know if the chase terminates on all databases for a rule set.

1. Use Critical Database
2. Run Chase  
**without blocked triggers**

$$\begin{aligned}x : \text{🏍️} &\rightarrow x : \text{⚙️} \\x : \text{⚙️} &\rightarrow x : \text{🌳🌳🌳} \vee x : \text{🏍️} \\x : \text{🌳🌳🌳} &\rightarrow f(x) : \text{🏍️} \wedge x \in f(x)\end{aligned}$$

Is  $x : \text{⚙️} \rightarrow x : \text{🌳🌳🌳} \vee x : \text{🏍️}$  with  $x/f(\star)$  blocked?

1. Backtrack facts for  $f(\star)$

$$f(\star) : \text{⚙️}$$



# How does Disjunctive MFA work?

We want to know if the chase terminates on all databases for a rule set.

1. Use Critical Database
2. Run Chase  
**without blocked triggers**

$$x : \text{🏍️} \rightarrow x : \text{⚙️}$$

$$x : \text{⚙️} \rightarrow x : \text{🌳🌳🌳} \vee x : \text{🏍️}$$

$$x : \text{🌳🌳🌳} \rightarrow f(x) : \text{🏍️} \wedge x \in f(x)$$

Is  $x : \text{⚙️} \rightarrow x : \text{🌳🌳🌳} \vee x : \text{🏍️}$  with  $x/f(\star)$  blocked?

1. Backtrack facts for  $f(\star)$

$$f(\star) : \text{⚙️} \quad f(\star) : \text{🏍️} \quad \star \in f(\star) \quad \star : \text{🌳🌳🌳}$$

# How does Disjunctive MFA work?

We want to know if the chase terminates on all databases for a rule set.

1. Use Critical Database
2. Run Chase  
**without blocked triggers**

$$\begin{aligned}x : \text{🏍️} &\rightarrow x : \text{⚙️} \\x : \text{⚙️} &\rightarrow x : \text{🌳🌳🌳} \vee x : \text{🏍️} \\x : \text{🌳🌳🌳} &\rightarrow f(x) : \text{🏍️} \wedge x \in f(x)\end{aligned}$$

Is  $x : \text{⚙️} \rightarrow x : \text{🌳🌳🌳} \vee x : \text{🏍️}$  with  $x/f(\star)$  blocked?

1. Backtrack facts for  $f(\star)$
2. Check Applicability

$$f(\star) : \text{⚙️} \quad f(\star) : \text{🏍️} \quad \star \in f(\star) \quad \star : \text{🌳🌳🌳}$$

# How does Disjunctive MFA work?

*We want to know if the chase terminates on all databases for a rule set.*

1. Use Critical Database
2. Run Chase  
**without blocked triggers**

**Lemma.** If a trigger is blocked it can never be applied in any chase tree.  
*Intuitively, because the backtracked facts must occur in every chase tree.*

Is  $x : \text{gear}$   $\rightarrow$   $x : \text{trees}$   $\vee$   $x : \text{motorcycle}$  with  $x/f(\star)$  blocked? **Yes!**

1. Backtrack facts for  $f(\star)$
2. Check Applicability

$f(\star) : \text{gear}$

$f(\star) : \text{motorcycle}$

$\star \in f(\star)$

$\star : \text{trees}$

# How does Disjunctive MFA work?

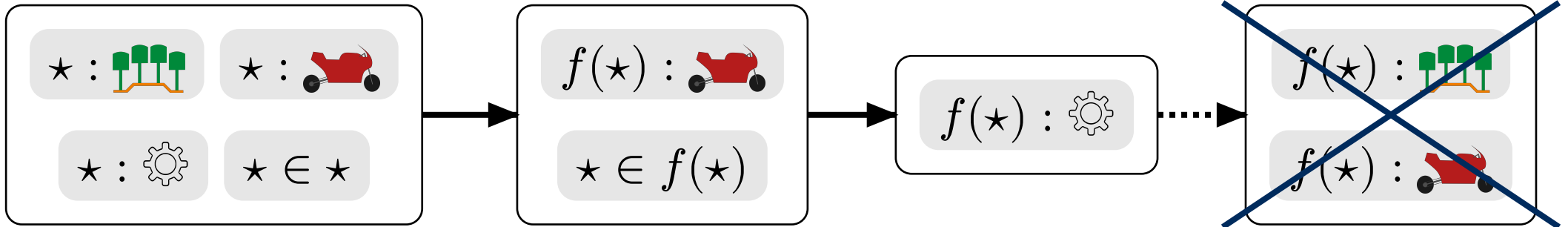
We want to know if the chase terminates on all databases for a rule set.

1. Use Critical Database
2. Run Chase  
**without blocked triggers**

$$x : \text{motorcycle} \rightarrow x : \text{gear}$$

$$x : \text{gear} \rightarrow x : \text{forest} \vee x : \text{motorcycle}$$

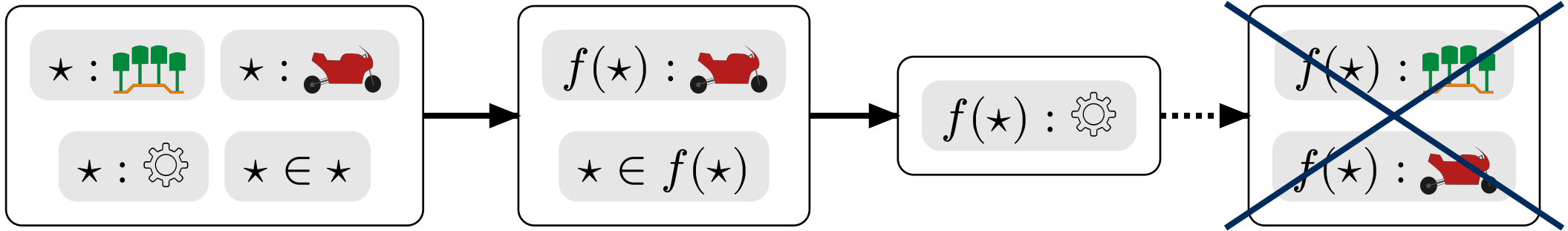
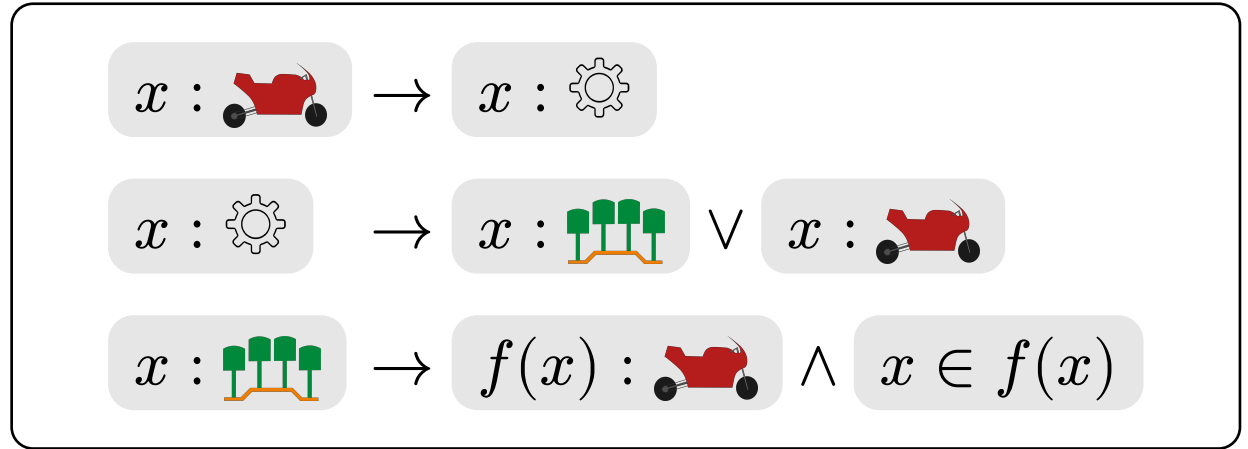
$$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$



# How does Disjunctive MFA work?

We want to know if the chase terminates on all databases for a rule set.

1. Use Critical Database
2. Run Chase  
**without blocked triggers**
3. Stop on Cyclic Term



# How does Disjunctive MFA work?

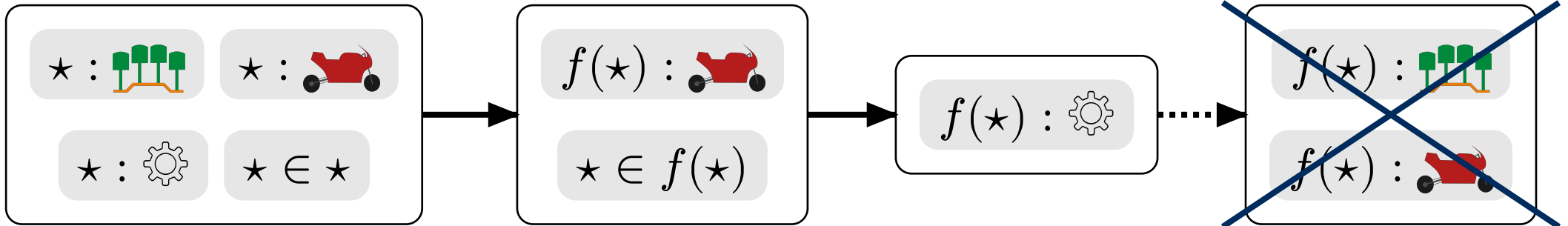
We want to know if the chase terminates on all databases for a rule set.

1. Use Critical Database
2. Run Chase  
**without blocked triggers**
3. Stop on Cyclic Term
4. DMFA iff no Cyclic Term

$$x : \text{motorcycle} \rightarrow x : \text{gear}$$

$$x : \text{gear} \rightarrow x : \text{forest} \vee x : \text{motorcycle}$$

$$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$

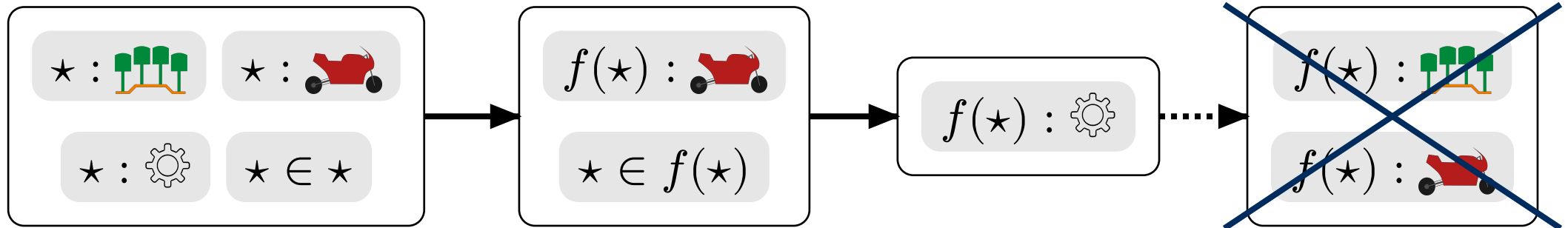


# How does Disjunctive MFA work?

*We want to know if the chase terminates on all databases for a rule set.*

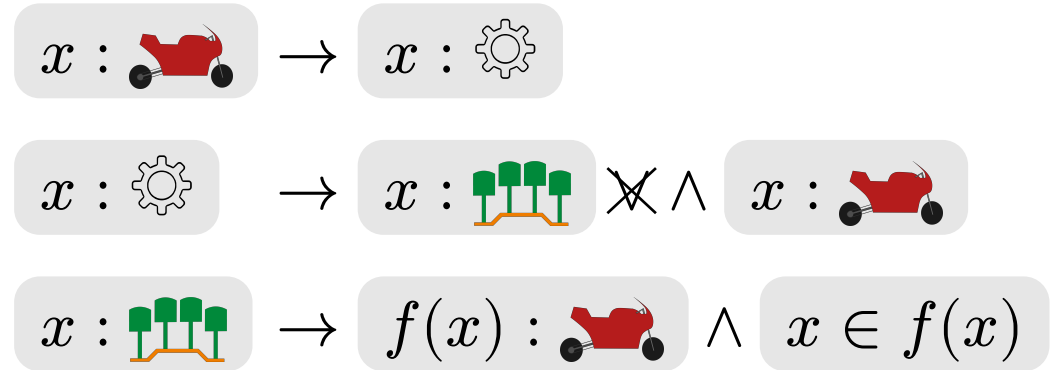
1. Use Critical Database
2. Run Chase  
**without blocked triggers**
3. Stop on Cyclic Term
4. DMFA iff no Cyclic Term

**Theorem.** If a rule set  $R$  is DMFA, then  $R$  terminates on every database. Again, the critical database chase subsumes every possible chase tree.



# How does Model Faithful Cyclicity work?

*We want to know if the chase always loops on some DB for a rule set.*

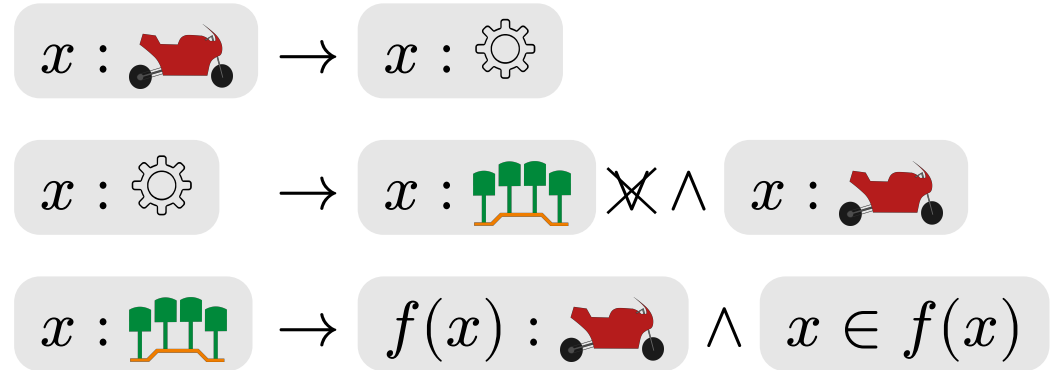




# How does Model Faithful Cyclicity work?

*We want to know if the chase always loops on some DB for a rule set.*

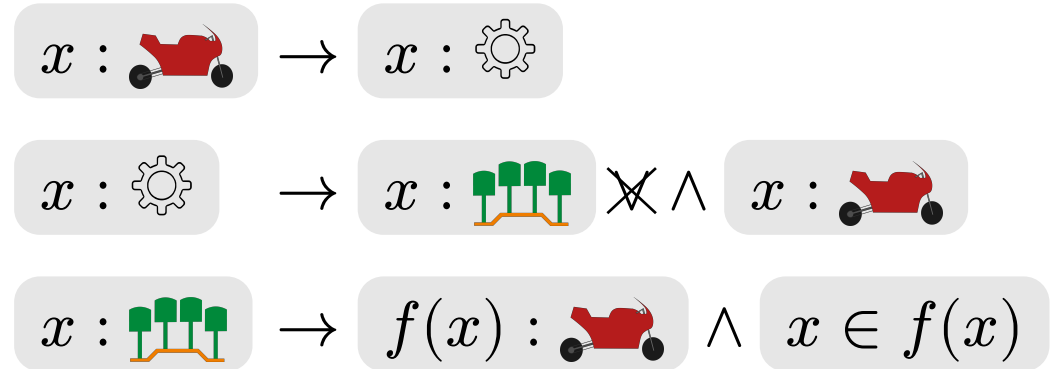
## 1. Ignore Disjunctive Rules



# How does Model Faithful Cyclicity work?

*We want to know if the chase always loops on some DB for a rule set.*

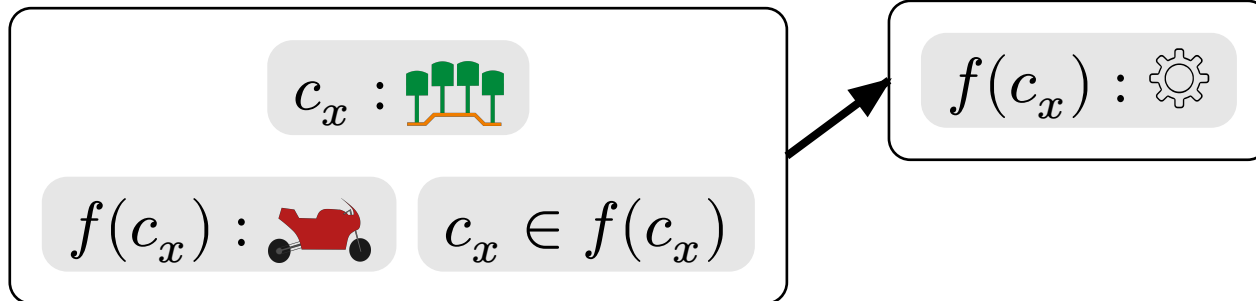
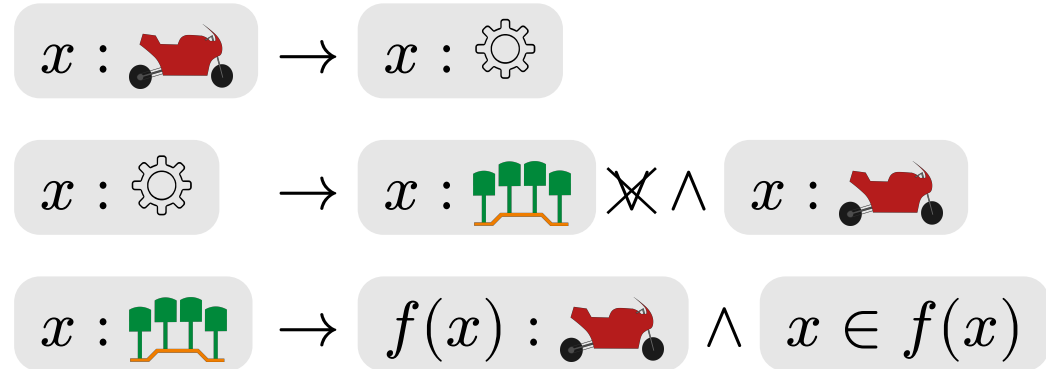
1. Ignore Disjunctive Rules
2. Use Rule Databases


$$c_x : \text{🌳}$$
$$f(c_x) : \text{🏍️} \quad c_x \in f(c_x)$$

# How does Model Faithful Cyclicity work?

*We want to know if the chase always loops on some DB for a rule set.*

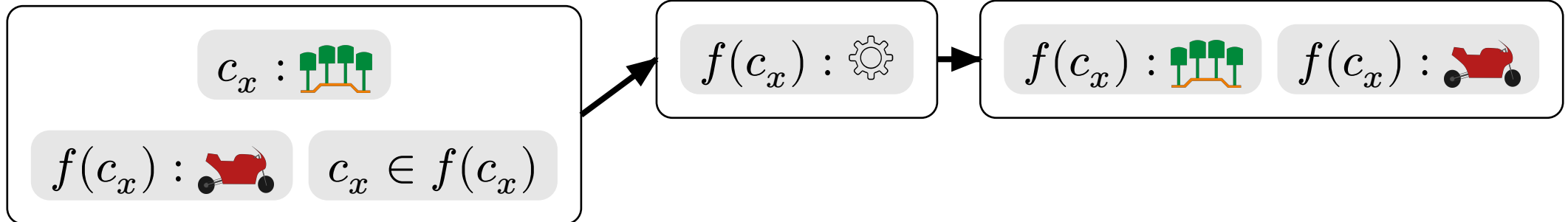
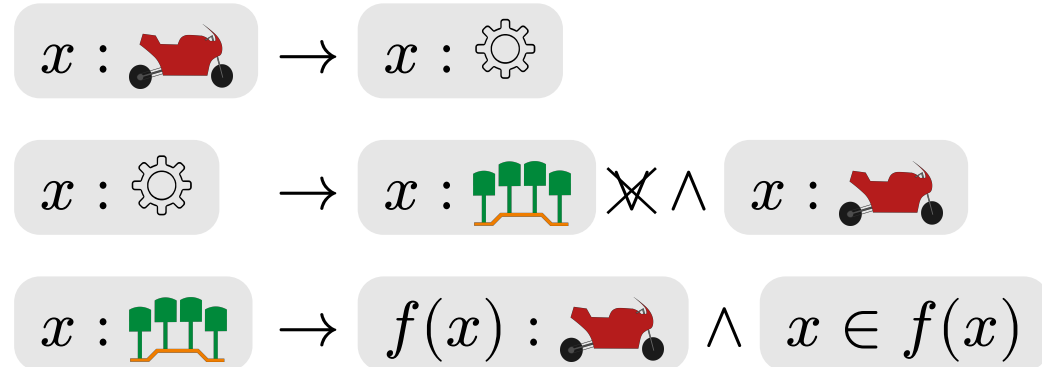
1. Ignore Disjunctive Rules
2. Use Rule Databases
3. Run Chase



# How does Model Faithful Cyclicity work?

*We want to know if the chase always loops on some DB for a rule set.*

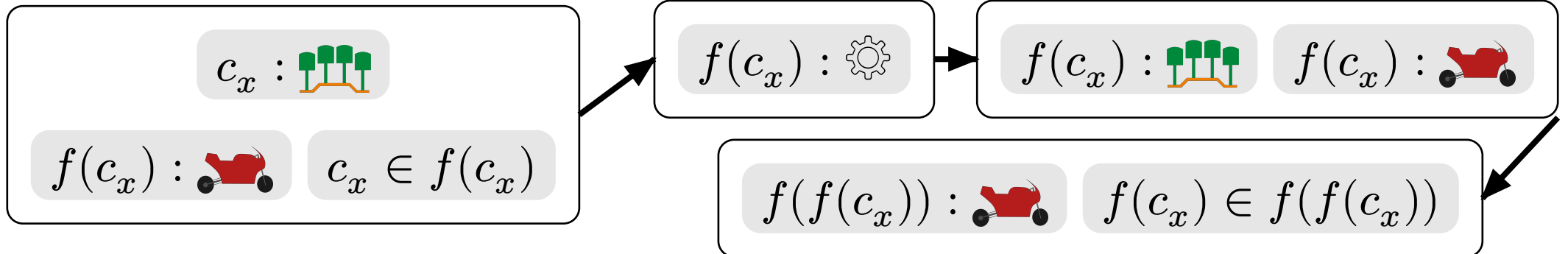
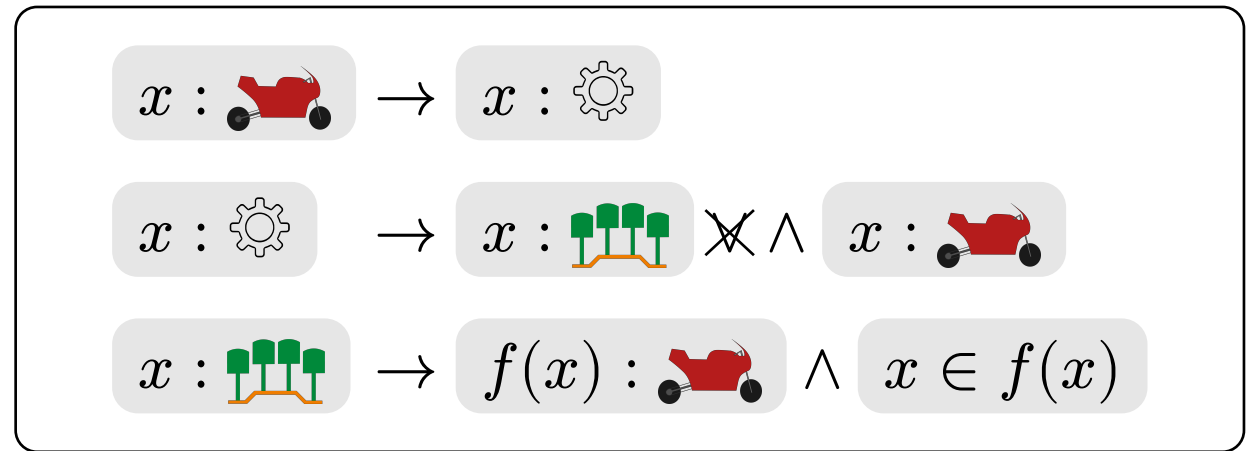
1. Ignore Disjunctive Rules
2. Use Rule Databases
3. Run Chase



# How does Model Faithful Cyclicity work?

*We want to know if the chase always loops on some DB for a rule set.*

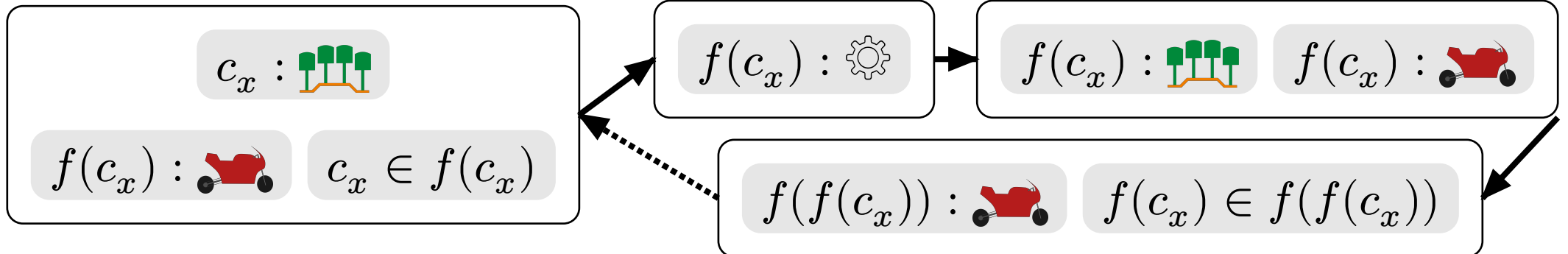
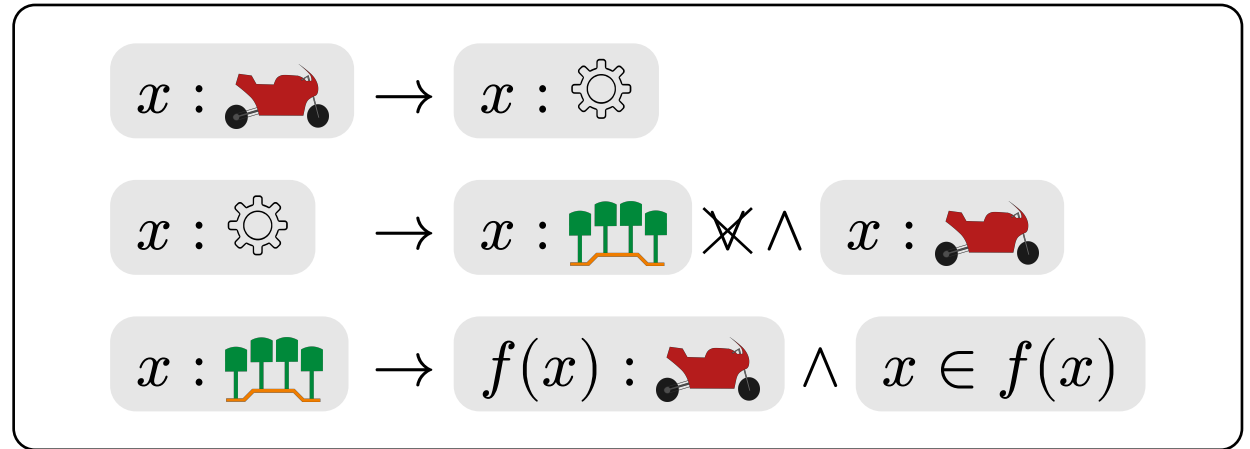
1. Ignore Disjunctive Rules
2. Use Rule Databases
3. Run Chase
4. Not beyond Cyclic Term



# How does Model Faithful Cyclicity work?

*We want to know if the chase always loops on some DB for a rule set.*

1. Ignore Disjunctive Rules
2. Use Rule Databases
3. Run Chase
4. Not beyond Cyclic Term
5. MFC iff Rule-Cyclic Term

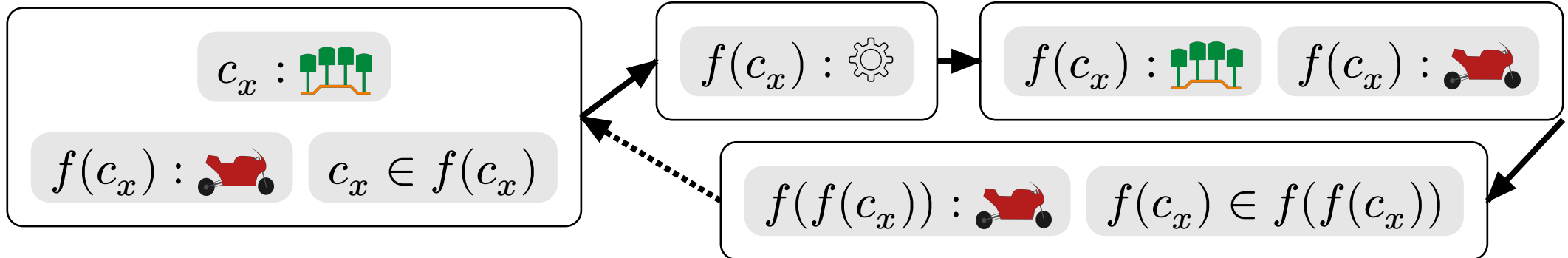


# How does Model Faithful Cyclicity work?

*We want to know if the chase always loops on some DB for a rule set.*

1. Ignore Disjunctive Rules
2. Use Rule Databases
3. Run Chase
4. Not beyond Cyclic Term
5. MFC iff Rule-Cyclic Term

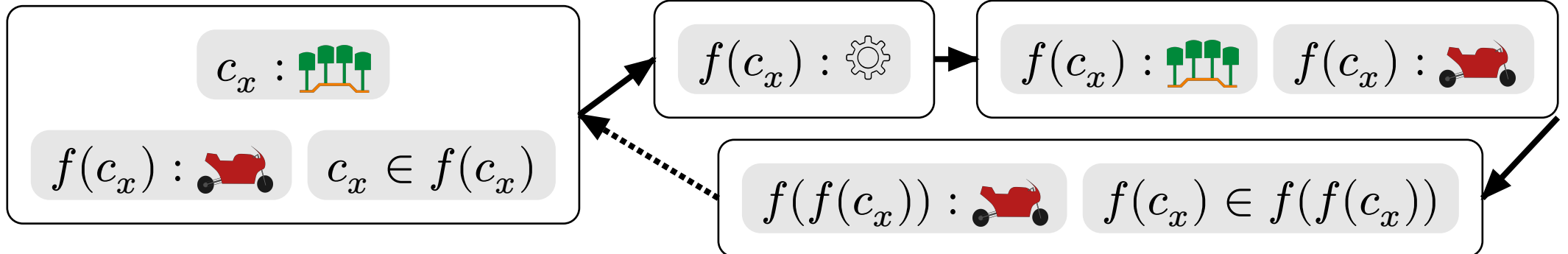
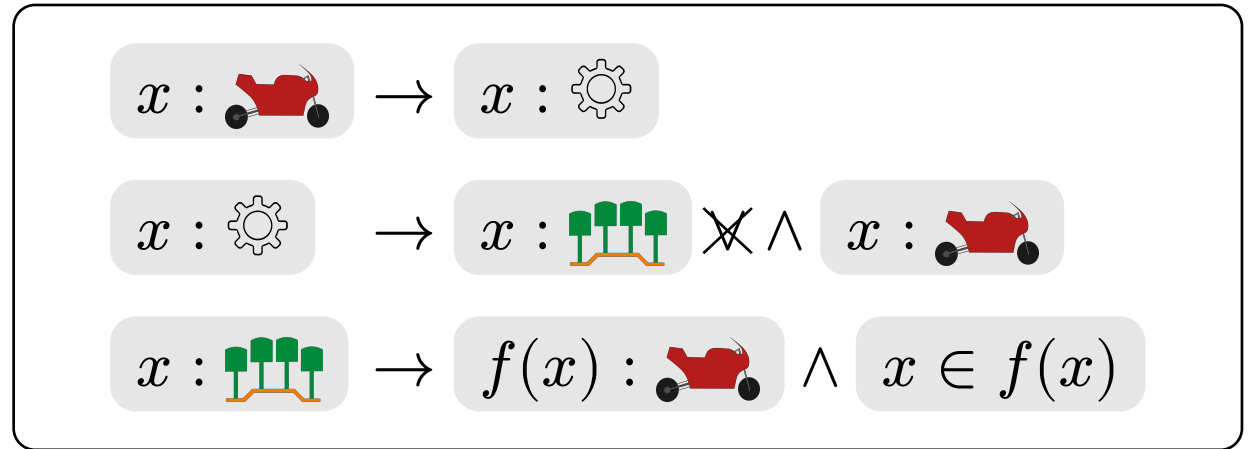
**Theorem.** If a rule set  $R$  is MFC, then there is a rule database for a rule in  $R$  that only admits infinite chase trees. *Intuitively, the loop occurs in every CT.*



# How does Model Faithful Cyclicity work?

*We want to know if the chase always loops on some DB for a rule set.*

1. Ignore Disjunctive Rules
2. Use Rule Databases
3. Run Chase
4. Not beyond Cyclic Term
5. MFC iff Rule-Cyclic Term





# How does Model Faithful Cyclicity work?

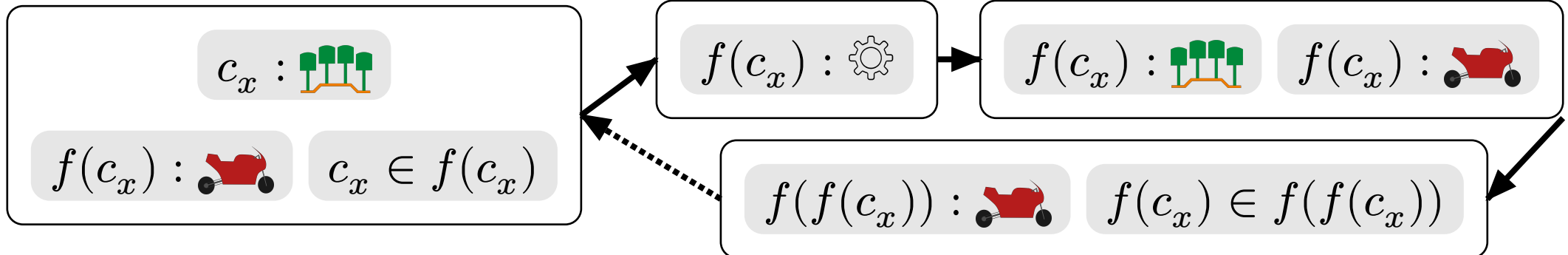
*We want to know if the chase always loops on some DB for a rule set.*

1. Ignore Disjunctive Rules
2. Use Rule Databases
3. Run Chase
4. Not beyond Cyclic Term
5. MFC iff Rule-Cyclic Term

$$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$$

$$x : \text{gear} \rightarrow x : \text{factory} \not\wedge x : \text{motorcycle}$$

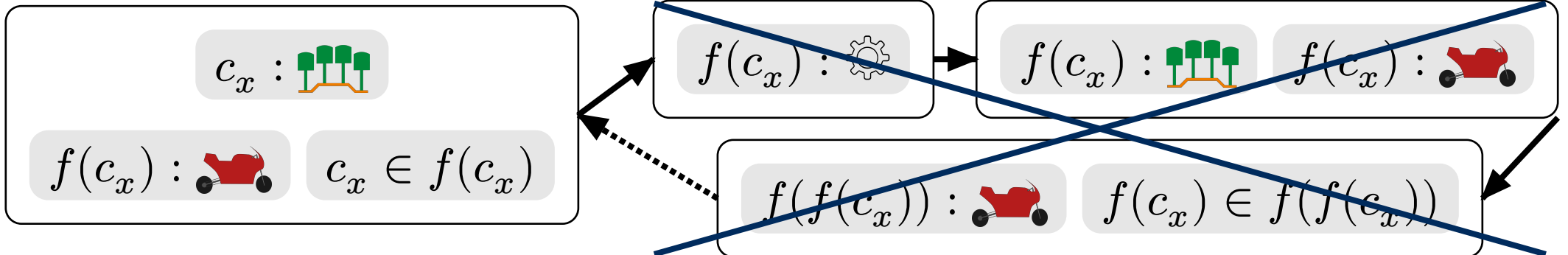
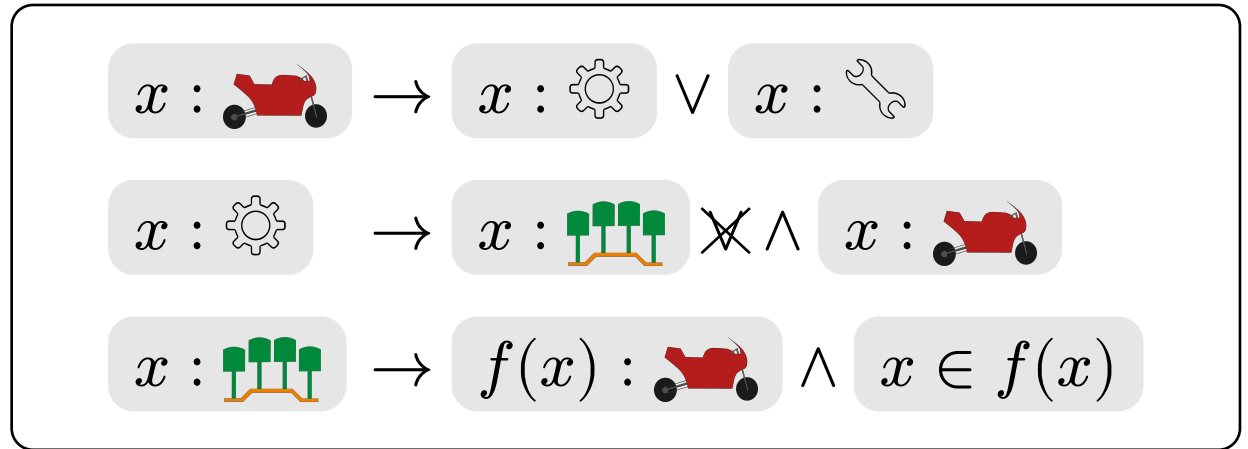
$$x : \text{factory} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$



# How does Model Faithful Cyclicity work?

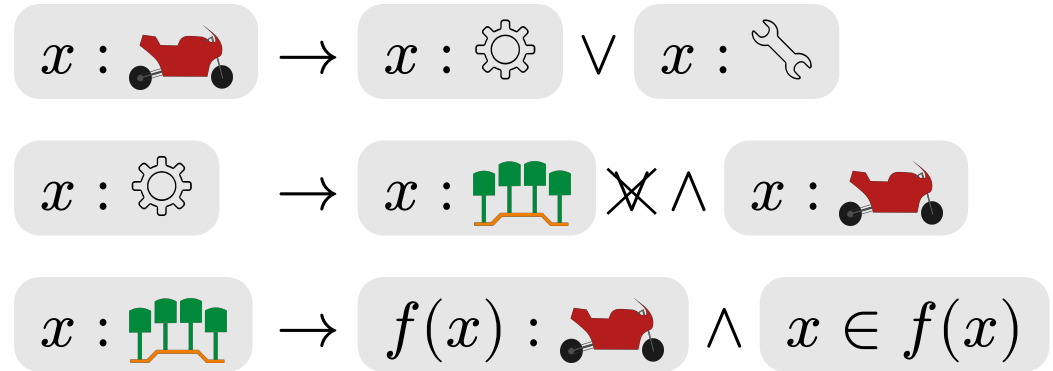
We want to know if the chase always loops on some DB for a rule set.

1. Ignore Disjunctive Rules
2. Use Rule Databases
3. Run Chase
4. Not beyond Cyclic Term
5. MFC iff Rule-Cyclic Term



# How does Disjunctive MFC work?

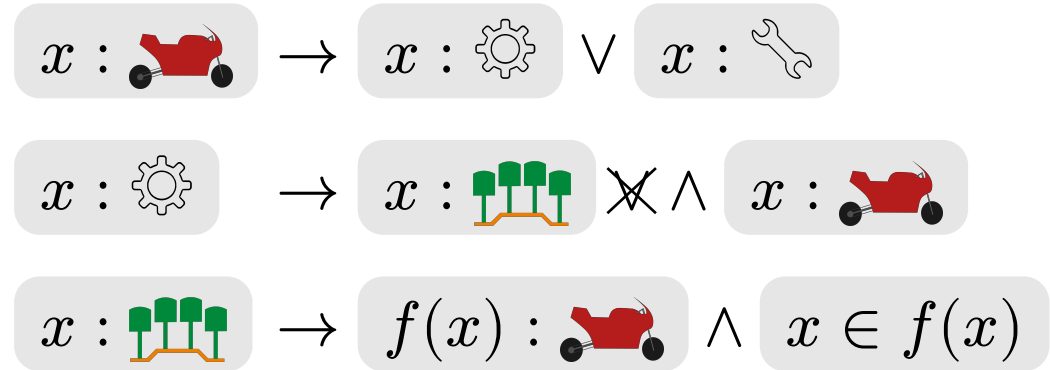
*We want to know if the chase always loops on some DB for a rule set.*



# How does Disjunctive MFC work?

*We want to know if the chase always loops on some DB for a rule set.*

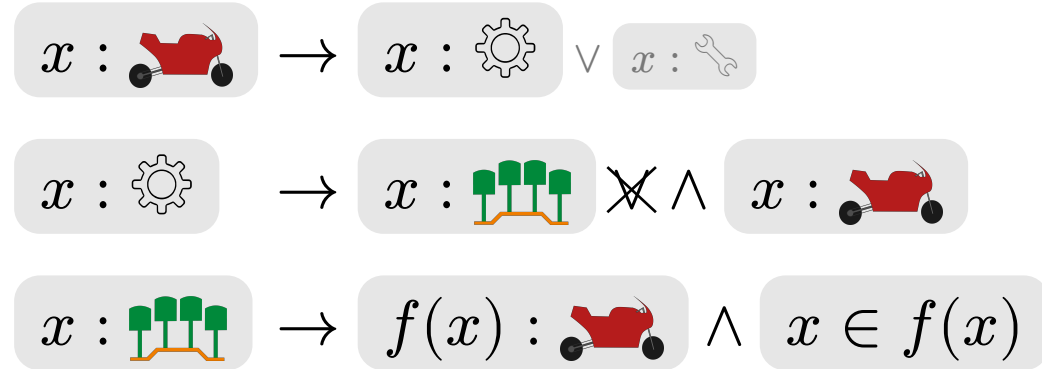
## 1. Ignore Disjunctive Rules



# How does Disjunctive MFC work?

*We want to know if the chase always loops on some DB for a rule set.*

## 1. Rule DB and Head-Choice



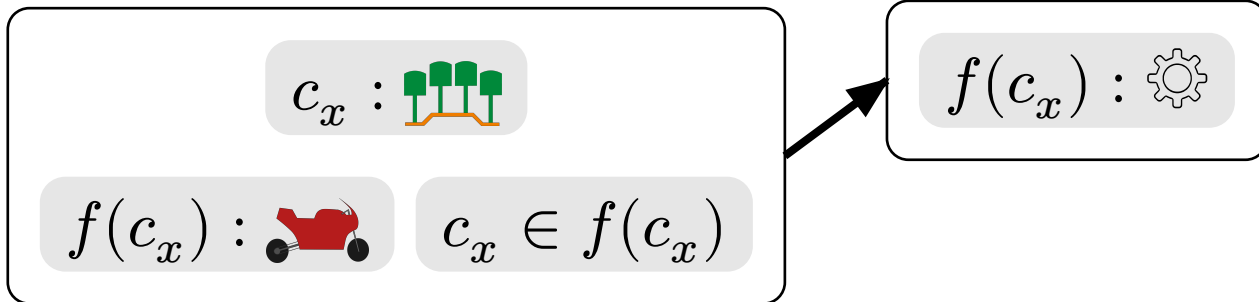
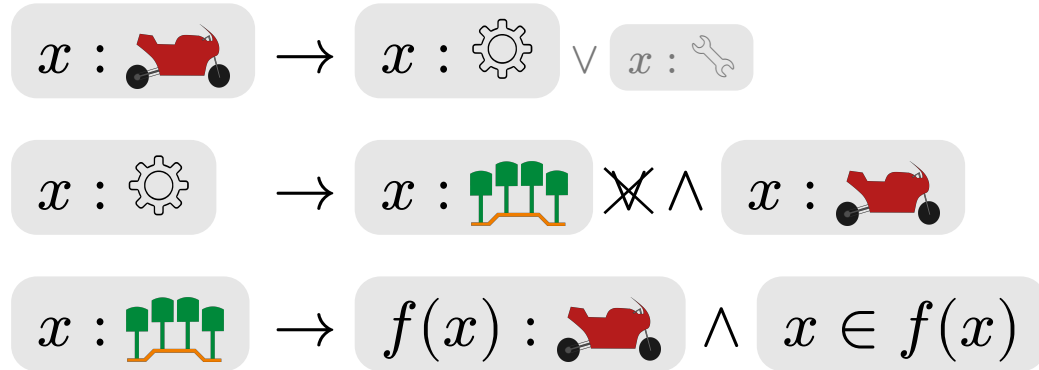
$c_x : \text{🌳}$

$f(c_x) : \text{🏍️}$     $c_x \in f(c_x)$

# How does Disjunctive MFC work?

*We want to know if the chase always loops on some DB for a rule set.*

1. Rule DB and Head-Choice
2. Run Chase  
**only unblockable trgs**



# How does Disjunctive MFC work?

We want to know if the chase always loops on some DB for a rule set.

1. Rule DB and Head-Choice
2. Run Chase  
**only unblockable trgs**

$$\begin{aligned}x : \text{motorcycle} &\rightarrow x : \text{gear} \vee x : \text{wrench} \\x : \text{gear} &\rightarrow x : \text{factory} \not\wedge x : \text{motorcycle} \\x : \text{factory} &\rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)\end{aligned}$$

Is  $x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$  with  $x/f(c_x)$  unblockable?

# How does Disjunctive MFC work?

*We want to know if the chase always loops on some DB for a rule set.*

1. Rule DB and Head-Choice
  2. Run Chase
- only unblockable trgs**

**Lemma.** If a trigger is unblockable, (1) its output must occur if its body is satisfied and (2) there are infinitely many *similar* unblockable triggers.

Is  $x : \text{🏍️} \rightarrow x : \text{⚙️} \vee x : \text{🔧}$  with  $x/f(c_x)$  unblockable?



# How does Disjunctive MFC work?

*We want to know if the chase always loops on some DB for a rule set.*

1. Rule DB and Head-Choice
2. Run Chase  
**only unblockable trgs**

**Lemma.** If a trigger is unblockable, (1) its output must occur if its body is satisfied and (2) there are infinitely many *similar* unblockable triggers.

Is  $x : \text{🏍️}$   $\rightarrow$   $x : \text{⚙️} \vee x : \text{🔧}$  with  $x/f(c_x)$  unblockable? **Yes!**  
*I will explain on the next slide.*

# How does Disjunctive MFC work?

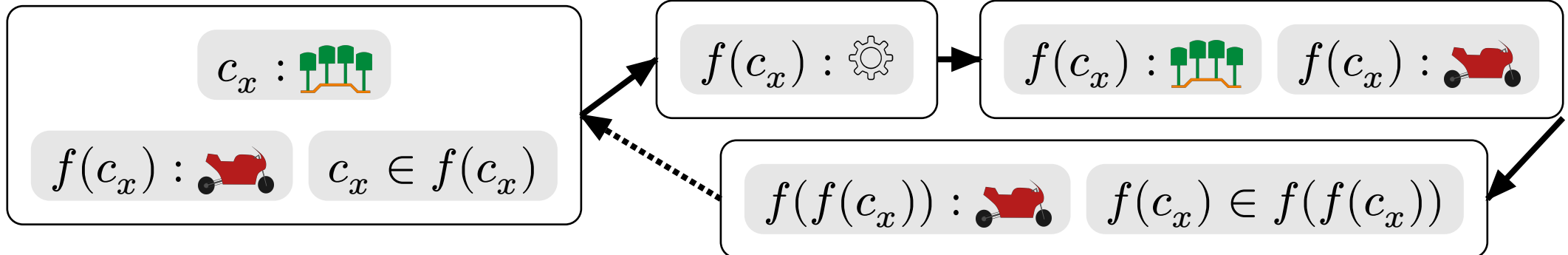
*We want to know if the chase always loops on some DB for a rule set.*

1. Rule DB and Head-Choice
2. Run Chase  
**only unblockable trgs**
3. Not beyond Cyclic Term
4. DMFC iff Rule-Cycl. Term

$$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$$

$$x : \text{gear} \rightarrow x : \text{factory} \not\wedge x : \text{motorcycle}$$

$$x : \text{factory} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$$

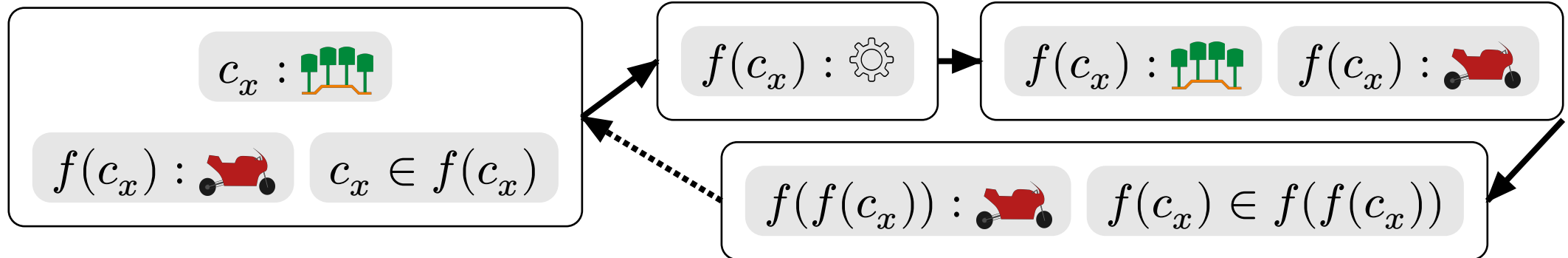


# How does Disjunctive MFC work?

*We want to know if the chase always loops on some DB for a rule set.*

1. Rule DB and Head-Choice
2. Run Chase  
**only unblockable trgs**
3. Not beyond Cyclic Term
4. DMFC iff Rule-Cycl. Term

**Theorem.** If a rule set  $R$  is DMFC, then there is a rule database for a rule in  $R$  that only admits infinite chase trees.  
*Infinite repetition of loop not obvious.*



$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$  with  $x/f(c_x)$  unblk?

*We compute an overapproximation of facts that can occur in the chase.*

$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$

$x : \text{gear} \rightarrow x : \text{forest} \not\wedge x : \text{motorcycle}$

$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$

$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$  with  $x / f(c_x)$  unblk?

We compute an overapproximation of facts that can occur in the chase.

1. Backtrack facts for  $f(c_x)$

$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$

$x : \text{gear} \rightarrow x : \text{forest} \not\wedge x : \text{motorcycle}$

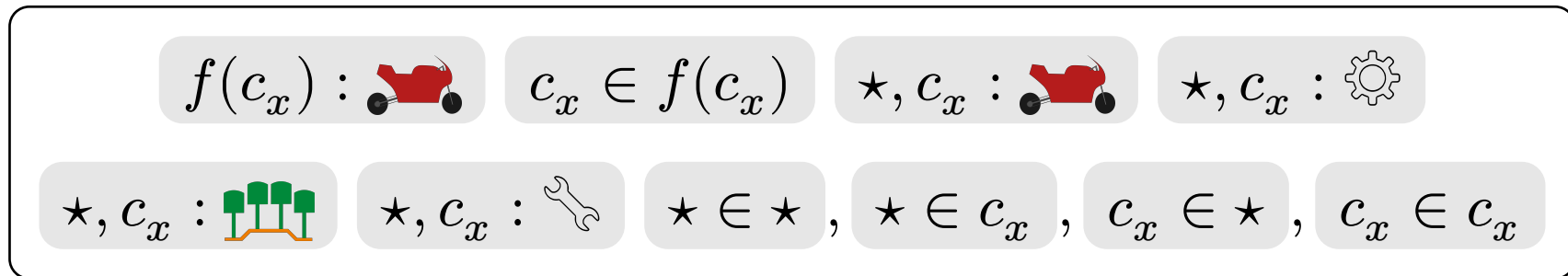
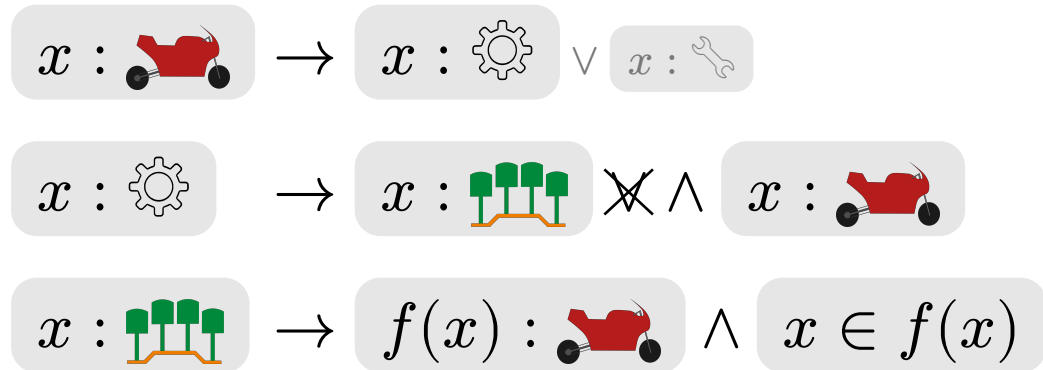
$x : \text{forest} \rightarrow f(x) : \text{motorcycle} \wedge x \in f(x)$

$f(c_x) : \text{motorcycle} \quad c_x \in f(c_x)$

$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$  with  $x/f(c_x)$  unblk?

We compute an overapproximation of facts that can occur in the chase.

1. Backtrack facts for  $f(c_x)$
2. Add critical facts



$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$  with  $x/f(c_x)$  unblk?

We compute an overapproximation of facts that can occur in the chase.

1. Backtrack facts for  $f(c_x)$
2. Add critical facts
3. Chase with Star-Rules except checked trigger

$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$

$x : \text{gear} \rightarrow x : \text{factory} \not\wedge x : \text{motorcycle}$

$x : \text{factory} \rightarrow \star : \text{motorcycle} \wedge x \in \star$

$f(c_x) : \text{motorcycle}$

$c_x \in f(c_x)$

$\star, c_x : \text{motorcycle}$

$\star, c_x : \text{gear}$

$\star, c_x : \text{factory}$

$\star, c_x : \text{wrench}$

$\star \in \star$

$\star \in c_x$

$c_x \in \star$

$c_x \in c_x$

$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$  with  $x/f(c_x)$  unblk?

We compute an overapproximation of facts that can occur in the chase.

1. Backtrack facts for  $f(c_x)$
2. Add critical facts
3. Chase with Star-Rules except checked trigger
4. Unblk iff applicable

$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$

$x : \text{gear} \rightarrow x : \text{factory} \not\wedge x : \text{motorcycle}$

$x : \text{factory} \rightarrow \star : \text{motorcycle} \wedge x \in \star$

$f(c_x) : \text{motorcycle}$

$c_x \in f(c_x)$

$\star, c_x : \text{motorcycle}$

$\star, c_x : \text{gear}$

$\star, c_x : \text{factory}$

$\star, c_x : \text{wrench}$

$\star \in \star$

$\star \in c_x$

$c_x \in \star$

$c_x \in c_x$

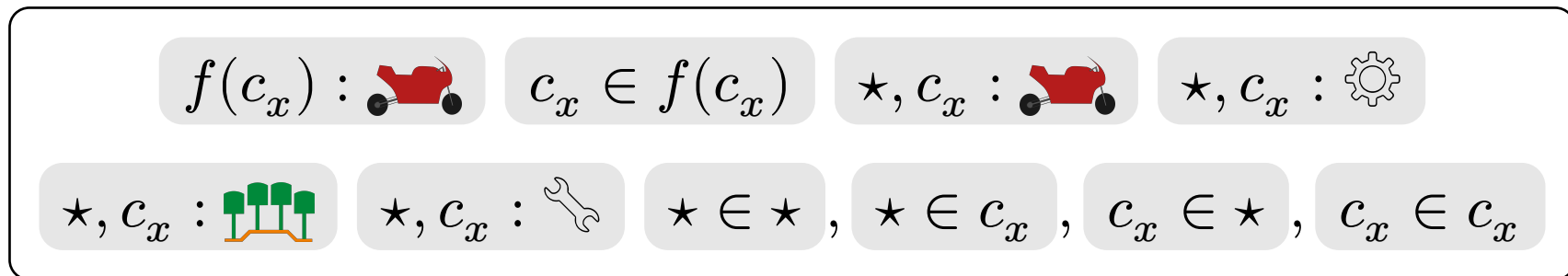


$x : \text{motorcycle} \rightarrow x : \text{gear} \vee x : \text{wrench}$  with  $x/f(c_x)$  unblk?

We compute an overapproximation of facts that can occur in the chase.

1. Backtrack facts for  $f(c_x)$
2. Add critical facts
3. Chase with Star-Rules except checked trigger
4. Unblk iff applicable

**Lemma.** If a trigger is unblockable, (1) its output must occur if its body is satisfied and (2) there are infinitely many *similar* unblockable triggers.



# Summary & Outlook

---

- We introduced **DMFA/DMFC** as sufficient conditions for **disjunctive skolem chase** (non-)termination based on ideas from **RMFA/RMFC**.

# Summary & Outlook

---

- 🚗 We introduced **DMFA/DMFC** as sufficient conditions for **disjunctive skolem chase** (non-)termination based on ideas from **RMFA/RMFC**.
- 🚗 We verified the generality of **DMFA/DMFC** in comparison to **MFA/MFC**.





# Summary & Outlook

---

- 🚗 We introduced **DMFA/DMFC** as sufficient conditions for **disjunctive skolem chase** (non-)termination based on ideas from **RMFA/RMFC**.
- 🚗 We verified the generality of **DMFA/DMFC** in comparison to **MFA/MFC**.
- ⚙️ In a follow-up work, we **ported non-termination improvements** and technicalities back to the **(disjunctive) restricted chase**. [GC23b]






# Summary & Outlook

---

-  We introduced **DMFA/DMFC** as sufficient conditions for **disjunctive skolem chase** (non-)termination based on ideas from **RMFA/RMFC**.
-  We verified the generality of **DMFA/DMFC** in comparison to **MFA/MFC**.
-  In a follow-up work, we **ported non-termination improvements** and technicalities back to the **(disjunctive) restricted chase**. [GC23b]
-  Investigate computation of the **disjunctive skolem chase with ASP**.







# Summary & Outlook

---

-  We introduced **DMFA/DMFC** as sufficient conditions for **disjunctive skolem chase** (non-)termination based on ideas from **RMFA/RMFC**.
-  We verified the generality of **DMFA/DMFC** in comparison to **MFA/MFC**.
-  In a follow-up work, we **ported non-termination improvements** and technicalities back to the **(disjunctive) restricted chase**. [GC23b]
-  Investigate computation of the **disjunctive skolem chase with ASP**.
-  Apply similar techniques in **other areas of KRR** (like ASP).

# Summary & Outlook

---

-  We introduced **DMFA/DMFC** as sufficient conditions for **disjunctive skolem chase** (non-)termination based on ideas from **RMFA/RMFC**.
-  We verified the generality of **DMFA/DMFC** in comparison to **MFA/MFC**.
-  In a follow-up work, we **ported non-termination improvements** and technicalities back to the **(disjunctive) restricted chase**. [GC23b]
-  Investigate computation of the **disjunctive skolem chase with ASP**.
-  Apply similar techniques in **other areas of KRR** (like ASP).
-  Look into **different translation techniques** from OWL into rules.

# Summary & Outlook

---

- 🚗 We introduced **DMFA/DMFC** as sufficient conditions for **disjunctive skolem chase** (non-)termination based on ideas from **RMFA/RMFC**.
- 🚗 We verified the generality of **DMFA/DMFC** in comparison to **MFA/MFC**.
- ⚙️ In a follow-up work, we **ported non-termination improvements** and technicalities back to the **(disjunctive) restricted chase**. [GC23b]
- 🔧 Investigate computation of the **disjunctive skolem chase with ASP**.
- 🔧 Apply similar techniques in **other areas of KRR** (like ASP).
- 🔧 Look into **different translation techniques** from OWL into rules.

*These slides are built using <https://typst.app> with the [Polylux](#) and [CeTZ](#) packages; give it a try! 😊*



# Summary & Outlook

---

- 🚗 We introduced **DMFA/DMFC** as sufficient conditions for **disjunctive skolem chase** (non-)termination based on ideas from **RMFA/RMFC**.
- 🚗 We verified the generality of **DMFA/DMFC** in comparison to **MFA/MFC**.
- ⚙️ In a follow-up work, we **ported non-termination improvements** and technicalities back to the **(disjunctive) restricted chase**. [GC23b]
- 🔧 Investigate computation of the **disjunctive skolem chase with ASP**.
- 🔧 Apply similar techniques in **other areas of KRR** (like ASP).
- 🔧 Look into **different translation techniques** from OWL into rules.

*These slides are built using <https://typst.app> with the [Polylux](#) and [CeTZ](#) packages; give it a try! 😎*

*(If you are fine with not having colored emojis in PDF exports yet. 😊)*

# References

---

- [Cue+13] B. Cuenca Grau, I. Horrocks, et al., “Acyclicity notions for existential rules and their application to query answering in ontologies,” *J. Artif. Intell. Resesearch (Jair)*, vol. 47, pp. 741–808, 2013.
- [GM14] T. Gogacz, and J. Marcinkowski, “All-instances termination of chase is undecidable,” in *Proc. 41st Int. Colloq. Automata, Languages, Program. (ICALP 2014), Denmark, Part II* in Lecture Notes in Computer Science, vol. 8573, 2014, pp. 293–304.
- [Bou+16] P. Bourhis, M. Manna, M. Morak, and A. Pieris, “Guarded-based disjunctive tuple-generating dependencies,” *ACM Trans. Database Syst. (Tods)*, vol. 41, no. 4, pp. 1–45, 2016.

- [CDK17] D. Carral, I. Dragoste, and M. Krötzsch, “Restricted chase (non)termination for existential rules with disjunctions,” in *Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI 2017), Aust.*, 2017, pp. 922–928.
- [G018] G. Grahne, and A. Onet, “Anatomy of the chase,” *Fundamenta Informaticae*, vol. 157, no. 3, pp. 221–270, 2018.
- [GC23a] L. Gerlach, and D. Carral, “General acyclicity and cyclicity notions for the disjunctive skolem chase,” *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 5, pp. 6372–6379, Jun. 2023A, doi: 10.1609/aaai.v37i5.25784. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/25784>
- [GC23b] L. Gerlach, and D. Carral, “Do repeat yourself: Understanding sufficient conditions for restricted chase non-termination,” in

*Proc. 20th Int. Conf. Princ. Knowl. Representation Reasoning*,  
2023B, pp. 301–310, doi: 10.24963/kr.2023/30. [Online]. Available:  
<https://doi.org/10.24963/kr.2023/30>

- [BV81] C. Beerli, and M. Y. Vardi, “The implication problem for data dependencies,” in *Proc. 8th Int. Colloq. Automata, Languages Program. (ICALP 1981)*, *Isr. in Lecture Notes in Computer Science*, vol. 115, 1981, pp. 73–85.