Computing Cores for Existential Rules with the Standard Chase and ASP

Markus Krötzsch
Knowledge-Based Systems, TU Dresden
Rules are simple, but what do they mean?

\[ R1 : \quad \text{father}(x, y) \rightarrow \text{male}(y) \]

\[ R2 : \quad \text{person}(x) \rightarrow \exists v. \text{father}(x, v) \land \text{male}(v) \]

\text{person(ada)} \quad \text{father(ada, george)}
Rules are simple, but what do they mean?

\[ R1 : \quad \text{father}(x, y) \rightarrow \text{male}(y) \]

\[ R2 : \quad \text{person}(x) \rightarrow \exists v. \text{father}(x, v) \land \text{male}(v) \]

\text{person(ada) \quad father(ada, george)}
Rules are simple, but what do they mean?

\[ R1 : \quad \text{father}(x, y) \rightarrow \text{male}(y) \]

\[ R2 : \quad \text{person}(x) \rightarrow \exists v. \text{father}(x, v) \land \text{male}(v) \]

\[
\begin{align*}
\text{person}(\text{ada}) & \quad \text{father}(\text{ada}, \text{george})
\end{align*}
\]
Rules are simple, but what do they mean?

\[ R1 : \quad \text{father}(x, y) \rightarrow \text{male}(y) \]
\[ R2 : \quad \text{person}(x) \rightarrow \exists v. \text{father}(x, v) \land \text{male}(v) \]

\[
\begin{align*}
\text{person(ada)} & \quad \text{father(ada, george)}
\end{align*}
\]
Rules are simple, but what do they mean?

\[ R1 : \quad \text{father}(x, y) \rightarrow \text{male}(y) \]

\[ R2 : \quad \text{person}(x) \rightarrow \exists v. \text{father}(x, v) \land \text{male}(v) \]

\[
\begin{align*}
\text{person}(\text{ada}) & \quad \text{father}(\text{ada, george}) \\
\end{align*}
\]
Rules are simple, but what do they mean?

\[ R1 : \text{father}(x, y) \rightarrow \text{male}(y) \]
\[ R2 : \text{person}(x) \rightarrow \exists v. \text{father}(x, v) \land \text{male}(v) \]

\text{person}(ada) \quad \text{father}(ada, george)
Rules are simple, but what do they mean?

\[ R1 : \text{father}(x, y) \rightarrow \text{male}(y) \]
\[ R2 : \text{person}(x) \rightarrow \exists v. \text{father}(x, v) \land \text{male}(v) \]

\text{person(ada)} \quad \text{father(ada, george)}
Simplification: We will only talk about finite chases here.

A **core** is a finite structure $C$ where every homomorphism $C \rightarrow C$ is an isomorphism.
Simplification: We will only talk about finite chases here.

A core is a finite structure $C$ where every homomorphism $C \rightarrow C$ is an isomorphism.
Cores in Practice

The core is the “best among all universal solutions”
– Fagin, Kolaitis, and Popa 2005

• Can be computed effectively
• Possible during the chase: “core chase”
The core is the “best among all universal solutions”
– Fagin, Kolaitis, and Popa 2005

• Can be computed effectively
• Possible during the chase: “core chase”

And yet: No current system implements the core chase!

**Problem:** Computing the core takes exponential time in the size of the chase.
Idea: Couldn’t we get cores with the standard chase?
Idea: Couldn’t we get cores with the standard chase?

Analysis: What went wrong here?

We applied rule $R_2$ to a match:

\[
\text{person}(\text{ada}) \rightarrow \text{father}(\text{ada}, \text{null}) \land \text{male}(\text{null})
\]

In the final chase, this instance is satisfied by an alternative match:

\[
\text{person}(\text{ada}) \rightarrow \text{father}(\text{ada}, \text{george}) \land \text{male}(\text{george})
\]
Cores from the Standard Chase

Idea: Couldn’t we get cores with the standard chase?

Analysis: What went wrong here?

- We applied rule $R2$ to a match:
  \[ \text{person(ada)} \rightarrow \text{father(ada, null)} \land \text{male(null)} \]
- In the final chase, this instance is satisfied by an alternative match:
  \[ \text{person(ada)} \rightarrow \text{father(ada, george)} \land \text{male(george)} \]
Cores from the Standard Chase

**Idea:** Couldn’t we get cores with the standard chase?

**Analysis:** What went wrong here?

- We applied rule $R2$ to a **match**:
  \[ \text{person(ada)} \rightarrow \text{father(ada, null) \land male(null)} \]
- In the final chase, this instance is satisfied by an **alternative match**:
  \[ \text{person(ada)} \rightarrow \text{father(ada, george) \land male(george)} \]

**Theorem:**
Every chase without alternative matches yields a core.
Idea: Characterise alternative-match-free standard chases in ASP.
A Characterisation in ASP

**Idea:** Characterise alternative-match-free standard chases in ASP.

**Encoding:**
- Use terms with (skolem) function symbols instead of named nulls
- Augment rules with precondition that they are “not blocked”
- Add rules that derive that a rule is “blocked” when an alternative match is found

Theorem: Cores from a chase without alternative matches correspond to the stable models of suitable normal logic programs.
Idea: Characterise alternative-match-free standard chases in ASP.

Encoding:
- Use terms with (skolem) function symbols instead of named nulls
- Augment rules with precondition that they are “not blocked”
- Add rules that derive that a rule is “blocked” when an alternative match is found

Theorem: Cores from a chase without alternative matches correspond to the stable models of suitable normal logic programs.
Can we guide the standard chase to produce a core?

Core Stratification:

- Define $R_1 \prec R_2$ to mean “$R_1$ could produce structures that enable alternative matches for $R_2$”
- Stratify the order of rule applications w.r.t. $\prec$ (together with a more usual positive “dependency” $\prec^+$)
Can we guide the standard chase to produce a core?

Core Stratification:

- Define $R_1 \prec^\square R_2$ to mean “$R_1$ could produce structures that enable alternative matches for $R_2$”
- Stratify the order of rule applications w.r.t. $\prec^\square$ (together with a more usual positive “dependency” $\prec^+$)

Results:

- Core stratification of a rule set can be decided in $\Sigma_2^P$.
- If a chase is core stratified, then it has no alternative matches (and therefore yields a core).
Finite universal model (=finite core)

Computed by a standard chase

Computed by chase without alternative matches

Core stratified chase

Polynomial core stratified chase
Existentials and Negation

A (classically) stratified logic program:

\[
\begin{align*}
R1 : & \quad father(x, y) \rightarrow male(y) \\
R2 : & \quad person(x) \rightarrow \exists v. \text{father}(x, v) \land \text{male}(v) \\
R3 : & \quad \text{father}(x, y) \rightarrow \text{equals}(y, y) \\
R4 : & \quad \text{father}(x, y_1) \land \text{father}(x, y_2) \land \\
& \quad \textbf{not} \ \text{equals}(y_1, y_2) \rightarrow \text{distinct}(y_1, y_2)
\end{align*}
\]
Existentials and Negation

A (classically) stratified logic program:

\[ R_1 : \text{father}(x, y) \rightarrow \text{male}(y) \]
\[ R_2 : \text{person}(x) \rightarrow \exists v. \text{father}(x, v) \land \text{male}(v) \]
\[ R_3 : \text{father}(x, y) \rightarrow \text{equals}(y, y) \]
\[ R_4 : \text{father}(x, y_1) \land \text{father}(x, y_2) \land \text{not equals}(y_1, y_2) \rightarrow \text{distinct}(y_1, y_2) \]
Perfect Core Models

**Idea:** Combine core stratification & classical stratification.

→ “Full stratification”
**Idea:** Combine core stratification & classical stratification.

~ “Full stratification”

**Theorem:** A finite, fully stratified chase yield a unique stable model that is a core, the perfect core model.
Main insight: Cores are in reach for practical uses

- Existing ASP engines can compute them
- Existing chase implementations can compute them
- Cores could be key to mix existentials and non-monotonic negation

Next questions:

- How do practical implementations perform?
- Is core stratification common in practice?
- Can we generalise perfect core models?