A tamed higher-arity extension of $\mathcal{ALC}$

Forward Guarded Fragment

June 22, 2021, QuantLA Research Seminar

Bartosz "Bart" Bednarczyk

TU Dresden & University of Wrocław

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them behaves nice, e.g. $\mathcal{ALC}$, $\mathcal{ALC}+\mathcal{H}$, $\mathcal{ALC}+\mathcal{Q}$ [Lutz'08]

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them behaves nice, e.g. $\mathcal{ALC}$, $\mathcal{ALC}+\mathcal{H}$, $\mathcal{ALC}+\mathcal{Q}$ [Lutz'08]

$\texttt{hasMother} \subseteq \texttt{hasParent}$

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them behaves nice, e.g. $\mathcal{ALC}$, $\mathcal{ALC}+\mathcal{H}$, $\mathcal{ALC}+\mathcal{Q}$ [Lutz'08]

$\mathtt{hasMother} \subseteq \mathtt{hasParent}$ $\quad$ $\mathit{Car} \sqsubseteq (=4).\mathtt{hasPart}\,\mathit{Wheel}$

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them behaves nice, e.g. $\mathcal{ALC}$, $\mathcal{ALC}+\mathcal{H}$, $\mathcal{ALC}+\mathcal{Q}$ [Lutz'08]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (= 4).\texttt{hasPart}\,\mathit{Wheel}$

Also with arithmetic and statistical properties [Baader, B., Rudolph'20]

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them behaves nice, e.g. $\mathcal{ALC}$, $\mathcal{ALC}+\mathcal{H}$, $\mathcal{ALC}+\mathcal{Q}$ [Lutz'08]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (= 4).\texttt{hasPart}\,\mathit{Wheel}$

Also with arithmetic and statistical properties [Baader, B., Rudolph'20]

As well as with regular expr, fixed points, (safe) role combination [B.'21, in prep.]

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them behaves nice, e.g. $\mathcal{ALC}$, $\mathcal{ALC}$+$\mathcal{H}$, $\mathcal{ALC}$+$\mathcal{Q}$ [Lutz'08]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (=4).\texttt{hasPart}\,\mathit{Wheel}$

Also with arithmetic and statistical properties [Baader, B., Rudolph'20]

As well as with regular expr, fixed points, (safe) role combination [B.'21, in prep.]

2. Some of them increase the complexity exponentially:

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them behaves nice, e.g. $\mathcal{ALC}$, $\mathcal{ALC}+\mathcal{H}$, $\mathcal{ALC}+\mathcal{Q}$ [Lutz'08]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (= 4).\texttt{hasPart}\,\mathit{Wheel}$

Also with arithmetic and statistical properties [Baader, B., Rudolph'20]

As well as with regular expr, fixed points, (safe) role combination [B.'21, in prep.]

2. Some of them increase the complexity exponentially:

E.g. transitivity [Eiter et al.'09], nominals (a.k.a. constants) [Ngo et al.'16]

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them behaves nice, e.g. $\mathcal{ALC}$, $\mathcal{ALC}+\mathcal{H}$, $\mathcal{ALC}+\mathcal{Q}$ [Lutz'08]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (=4).\texttt{hasPart}\,\mathit{Wheel}$

Also with arithmetic and statistical properties [Baader, B., Rudolph'20]

As well as with regular expr, fixed points, (safe) role combination [B.'21, in prep.]

2. Some of them increase the complexity exponentially:

E.g. transitivity [Eiter et al.'09], nominals (a.k.a. constants) [Ngo et al.'16]

more: inverses [Lutz'07], self-loops [B., Rudolph'21 Submitted.]

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them behaves nice, e.g. $\mathcal{ALC}$, $\mathcal{ALC}+\mathcal{H}$, $\mathcal{ALC}+\mathcal{Q}$ [Lutz'08]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (=4).\texttt{hasPart}\,\mathit{Wheel}$

Also with arithmetic and statistical properties [Baader, B., Rudolph'20]

As well as with regular expr, fixed points, (safe) role combination [B.'21, in prep.]

2. Some of them increase the complexity exponentially:

E.g. transitivity [Eiter et al.'09], nominals (a.k.a. constants) [Ngo et al.'16]

more: inverses [Lutz'07], self-loops [B., Rudolph'21 Submitted.]

What makes $\mathcal{ALC}$ easy, but $\mathcal{ALCI}$ and the others hard?

Our motivation: … ng hard for $\mathcal{ALC}$?

1. Some of them beh… $+\mathcal{Q}$ [Lutz'08]

`hasMother` $\subseteq$ `hasPar`… `Part` *Wheel*

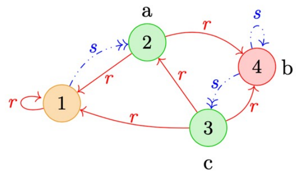
Also with arithmetic ar… udolph'20]

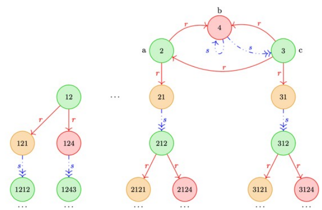
As well as with regular … ation [B.'21, in prep.]

2. Some of them incr…

E.g. transitivity [Eiter … [Ngo et al.'16]

more: inverses [Lutz'07 … ed.]





What makes $\mathcal{ALC}$ easy, but $\mathcal{ALCI}$ and the others hard?

Answer: Forward models!

Our motivation: … ng hard for $\mathcal{ALC}$?

1. Some of them beh… +$\mathcal{Q}$ [Lutz'08]

`hasMother` $\subseteq$ `hasPar` … `Part` *Wheel*

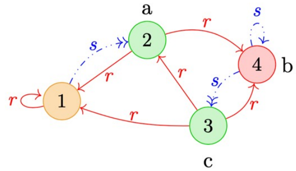
Also with arithmetic ar … udolph'20]

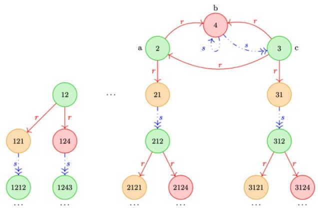
As well as with regular … ation [B.'21, in prep.]

2. Some of them incr…

E.g. transitivity [Eiter … [Ngo et al.'16]

more: inverses [Lutz'07 … ed.]





What makes $\mathcal{ALC}$ easy, but $\mathcal{ALCI}$ and the others hard?

Answer: Forward models!

Can we find a higher-arity version of $\mathcal{ALC}$ with ExpTime querying?

Our motivation: ... ng hard for $\mathcal{ALC}$?

1. Some of them beh ... $+\mathcal{Q}$ [Lutz'08]

`hasMother` $\subseteq$ `hasPar` ... `Part` *Wheel*

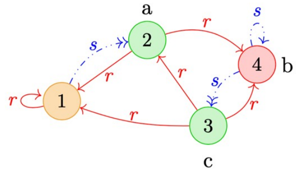
Also with arithmetic ar ... udolph'20]

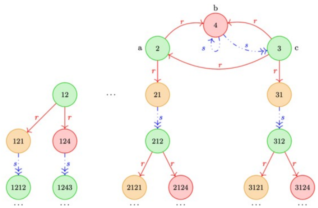
As well as with regular ... ation [B.'21, in prep.]

2. Some of them incr

E.g. transitivity [Eiter ... [Ngo et al.'16]

more: inverses [Lutz'07 ... ed.]





What makes $\mathcal{ALC}$ easy, but $\mathcal{ALCI}$ and the others hard?

Answer: Forward models!

Can we find a higher-arity version of $\mathcal{ALC}$ with ExpTime querying?

Yes! $\mathcal{FGF}$ [B. JELIA'21, This talk!]

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The guarded fragment of $\mathcal{FO}$ is obtained by relativising quantifiers by atoms.

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The guarded fragment of $\mathcal{FO}$ is obtained by relativising quantifiers by atoms.
- $\exists\vec{y}\ \alpha(\vec{x},\vec{y})\wedge\varphi(\vec{x},\vec{y}), \forall\vec{y}\ \alpha(\vec{x},\vec{y})\rightarrow\varphi(\vec{x},\vec{y})$ – guard must cover free variables of φ.

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The guarded fragment of $\mathcal{FO}$ is obtained by relativising quantifiers by atoms.
- $\exists \vec{y}\ \alpha(\vec{x}, \vec{y}) \wedge \varphi(\vec{x}, \vec{y}), \forall \vec{y}\ \alpha(\vec{x}, \vec{y}) \rightarrow \varphi(\vec{x}, \vec{y})$ – guard must cover free variables of φ.

Example 1. Some artist admires only beekeepers

$$\exists x\ artst(x) \wedge \forall y\ \left(adm(x, y) \rightarrow bkpr(y)\right)$$

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The guarded fragment of $\mathcal{FO}$ is obtained by relativising quantifiers by atoms.
- $\exists \vec{y}\ \alpha(\vec{x}, \vec{y}) \wedge \varphi(\vec{x}, \vec{y}), \forall \vec{y}\ \alpha(\vec{x}, \vec{y}) \rightarrow \varphi(\vec{x}, \vec{y})$ – guard must cover free variables of φ.

Example 1. Some artist admires only beekeepers

$$\exists x\ artst(x) \wedge \forall y\ \ (adm(x, y) \rightarrow bkpr(y))$$

Example 2. Every artist envies every bekeeper he admires

$$\forall x\ artst(x) \rightarrow \forall y\ [adm(x, y) \rightarrow (bkpr(y) \rightarrow env(x, y))]$$

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The guarded fragment of $\mathcal{FO}$ is obtained by relativising quantifiers by atoms.
- $\exists \vec{y}\ \alpha(\vec{x}, \vec{y}) \wedge \varphi(\vec{x}, \vec{y}), \forall \vec{y}\ \alpha(\vec{x}, \vec{y}) \rightarrow \varphi(\vec{x}, \vec{y})$ – guard must cover free variables of φ.

Example 1. Some artist admires only beekeepers

$$\exists x\ artst(x) \wedge \forall y\ (adm(x, y) \rightarrow bkpr(y))$$

Example 2. Every artist envies every bekeeper he admires

$$\forall x\ artst(x) \rightarrow \forall y\ [adm(x, y) \rightarrow (bkpr(y) \rightarrow env(x, y))]$$

Coexample 3. Every artist admires every beekeeper

$$\forall x\ (artst(x) \rightarrow \forall y\ (bkpr(y) \rightarrow adm(x, y)))$$

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The guarded fragment of $\mathcal{FO}$ is obtained by relativising quantifiers by atoms.
- $\exists \vec{y}\ \alpha(\vec{x}, \vec{y}) \wedge \varphi(\vec{x}, \vec{y}), \forall \vec{y}\ \alpha(\vec{x}, \vec{y}) \rightarrow \varphi(\vec{x}, \vec{y})$ – guard must cover free variables of φ.

Example 1. Some artist admires only beekeepers

$$\exists x\ artst(x) \wedge \forall y\ (adm(x, y) \rightarrow bkpr(y))$$

Example 2. Every artist envies every bekeeper he admires

$$\forall x\ artst(x) \rightarrow \forall y\ [adm(x, y) \rightarrow (bkpr(y) \rightarrow env(x, y))]$$

Coexample 3. Every artist admires every beekeeper

$$\forall x\ (artst(x) \rightarrow \forall y\ (bkpr(y) \rightarrow adm(x, y)))$$

Theorem (Grädel 1999)

The satisfiability problem for $\mathcal{GF}$ is $2\textsc{ExpTime}$-complete.

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The guarded fragment of $\mathcal{FO}$ is obtained by relativising quantifiers by atoms.
- $\exists \vec{y}\ \alpha(\vec{x}, \vec{y}) \wedge \varphi(\vec{x}, \vec{y}), \forall \vec{y}\ \alpha(\vec{x}, \vec{y}) \rightarrow \varphi(\vec{x}, \vec{y})$ – guard must cover free variables of φ.

Example 1. Some artist admires only beekeepers

$$\exists x\ artst(x) \wedge \forall y\ (adm(x, y) \rightarrow bkpr(y))$$

Example 2. Every artist envies every bekeeper he admires

$$\forall x\ artst(x) \rightarrow \forall y\ [adm(x, y) \rightarrow (bkpr(y) \rightarrow env(x, y))]$$

Coexample 3. Every artist admires every beekeeper

$$\forall x\ (artst(x) \rightarrow \forall y\ (bkpr(y) \rightarrow adm(x, y)))$$

Theorem (Grädel 1999)

The satisfiability problem for $\mathcal{GF}$ is $2\textsc{ExpTime}$-complete.

Theorem (Bárány et al. 2013)

Conjunctive query entailment problem for $\mathcal{GF}$ is $2\textsc{ExpTime}$-complete.

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The fluted fragment of $\mathcal{FO}$ is obtained by keeping the variables ordered.

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The fluted fragment of $\mathcal{FO}$ is obtained by keeping the variables ordered.
- In atoms we can use only suffixes of the sequences of already quantified variables.

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The fluted fragment of $\mathcal{FO}$ is obtained by keeping the variables ordered.
- In atoms we can use only suffixes of the sequences of already quantified variables.

Example 1. No student admires every professor

$$\forall x_1(\mathit{stud}(x_1) \rightarrow \neg\forall x_2(\mathit{prof}(x_2) \rightarrow \mathit{admires}(x_1, x_2)))$$

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The fluted fragment of $\mathcal{FO}$ is obtained by keeping the variables ordered.
- In atoms we can use only suffixes of the sequences of already quantified variables.

Example 1. No student admires every professor

$$\forall x_1(\mathit{stud}(x_1) \rightarrow \neg\forall x_2(\mathit{prof}(x_2) \rightarrow \mathit{admires}(x_1, x_2)))$$

Example 2. No lecturer introduces any professor to every student

$$\forall x_1(\mathit{lect}(x_1) \rightarrow \neg\exists x_2(\mathit{prof}(x_2) \wedge \forall x_3(\mathit{stud}(x_3) \rightarrow \mathit{intro}(x_1, x_2, x_3))))$$

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The fluted fragment of $\mathcal{FO}$ is obtained by keeping the variables ordered.
- In atoms we can use only suffixes of the sequences of already quantified variables.

Example 1. No student admires every professor

$$\forall x_1(stud(x_1) \rightarrow \neg\forall x_2(prof(x_2) \rightarrow admires(x_1, x_2)))$$

Example 2. No lecturer introduces any professor to every student

$$\forall x_1(lect(x_1) \rightarrow \neg\exists x_2(prof(x_2) \wedge \forall x_3(stud(x_3) \rightarrow intro(x_1, x_2, x_3))))$$

Coexample 1. $\forall x_1 r(x_1, x_1)$

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The fluted fragment of $\mathcal{FO}$ is obtained by keeping the variables ordered.
- In atoms we can use only suffixes of the sequences of already quantified variables.

Example 1. No student admires every professor

$$\forall x_1 (stud(x_1) \rightarrow \neg \forall x_2 (prof(x_2) \rightarrow admires(x_1, x_2)))$$

Example 2. No lecturer introduces any professor to every student

$$\forall x_1 (lect(x_1) \rightarrow \neg \exists x_2 (prof(x_2) \land \forall x_3 (stud(x_3) \rightarrow intro(x_1, x_2, x_3))))$$

Coexample 1. $\forall x_1 r(x_1, x_1)$

Coexample 2. $\forall x_1 \forall x_2 r(x_1, x_2) \rightarrow s(x_2, x_1)$

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The fluted fragment of $\mathcal{FO}$ is obtained by keeping the variables ordered.
- In atoms we can use only suffixes of the sequences of already quantified variables.

Example 1. No student admires every professor

$$\forall x_1 (stud(x_1) \to \neg \forall x_2 (prof(x_2) \to admires(x_1, x_2)))$$

Example 2. No lecturer introduces any professor to every student

$$\forall x_1 (lect(x_1) \to \neg \exists x_2 (prof(x_2) \wedge \forall x_3 (stud(x_3) \to intro(x_1, x_2, x_3))))$$

Coexample 1. $\forall x_1 r(x_1, x_1)$

Coexample 2. $\forall x_1 \forall x_2 r(x_1, x_2) \to s(x_2, x_1)$

Coexample 3. $\forall x_1 \forall x_2 \forall x_3 r(x_1, x_2) \wedge r(x_2, x_3) \to r(x_1, x_3)$

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The fluted fragment of $\mathcal{FO}$ is obtained by keeping the variables ordered.
- In atoms we can use only suffixes of the sequences of already quantified variables.

Example 1. No student admires every professor

$$\forall x_1(\mathit{stud}(x_1) \rightarrow \neg\forall x_2(\mathit{prof}(x_2) \rightarrow \mathit{admires}(x_1, x_2)))$$

Example 2. No lecturer introduces any professor to every student

$$\forall x_1(\mathit{lect}(x_1) \rightarrow \neg\exists x_2(\mathit{prof}(x_2) \wedge \forall x_3(\mathit{stud}(x_3) \rightarrow \mathit{intro}(x_1, x_2, x_3))))$$

Coexample 1. $\forall x_1 r(x_1, x_1)$

Coexample 2. $\forall x_1 \forall x_2 r(x_1, x_2) \rightarrow s(x_2, x_1)$

Coexample 3. $\forall x_1 \forall x_2 \forall x_3 r(x_1, x_2) \wedge r(x_2, x_3) \rightarrow r(x_1, x_3)$

Theorem (Pratt-Hartman et al. 2016)

The satisfiability problem for $\mathcal{FL}$ is $\textsc{Tower}$-complete.

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The fluted fragment of $\mathcal{FO}$ is obtained by keeping the variables ordered.
- In atoms we can use only suffixes of the sequences of already quantified variables.

Example 1. No student admires every professor

$$\forall x_1 (stud(x_1) \to \neg \forall x_2 (prof(x_2) \to admires(x_1, x_2)))$$

Example 2. No lecturer introduces any professor to every student

$$\forall x_1 (lect(x_1) \to \neg \exists x_2 (prof(x_2) \wedge \forall x_3 (stud(x_3) \to intro(x_1, x_2, x_3))))$$

Coexample 1. $\forall x_1 r(x_1, x_1)$

Coexample 2. $\forall x_1 \forall x_2 r(x_1, x_2) \to s(x_2, x_1)$

Coexample 3. $\forall x_1 \forall x_2 \forall x_3 r(x_1, x_2) \wedge r(x_2, x_3) \to r(x_1, x_3)$

Theorem (Pratt-Hartman et al. 2016)

The satisfiability problem for $\mathcal{FL}$ is $\textsc{Tower}$-complete.

If we replace suffices by infixes in $\mathcal{FL}$ we get the forward fragment $\mathcal{FF}$.

Two nice logics: $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

- The fluted fragment of $\mathcal{FO}$ is obtained by keeping the variables ordered.
- In atoms we can use only suffixes of the sequences of already quantified variables.

Example 1. No student admires every professor

$$\forall x_1 (stud(x_1) \to \neg \forall x_2 (prof(x_2) \to admires(x_1, x_2)))$$

Example 2. No lecturer introduces any professor to every student

$$\forall x_1 (lect(x_1) \to \neg \exists x_2 (prof(x_2) \land \forall x_3 (stud(x_3) \to intro(x_1, x_2, x_3))))$$

Coexample 1. $\forall x_1 r(x_1, x_1)$

Coexample 2. $\forall x_1 \forall x_2 r(x_1, x_2) \to s(x_2, x_1)$

Coexample 3. $\forall x_1 \forall x_2 \forall x_3 r(x_1, x_2) \land r(x_2, x_3) \to r(x_1, x_3)$

Theorem (Pratt-Hartman et al. 2016)

The satisfiability problem for $\mathcal{FL}$ is $\textsc{Tower}$-complete.

If we replace suffices by infixes in $\mathcal{FL}$ we get the forward fragment $\mathcal{FF}$.

Lemma (B. 2021)

$\mathcal{FF}$ is reducible to $\mathcal{FL}$ in polynomial time.

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Both $\mathcal{GF}$ and $\mathcal{FF}$ capture $\mathcal{ALC}$, e.g.: "Grandfathers with granddaughters"

$$\texttt{grf-wth-gdtrs} \sqsubseteq \exists\texttt{hasChld}.\exists\texttt{hasChld}.\mathit{female}$$

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Both $\mathcal{GF}$ and $\mathcal{FF}$ capture $\mathcal{ALC}$, e.g.: "Grandfathers with granddaughters"

$$\texttt{grf-wth-gdtrs} \sqsubseteq \exists \texttt{hasChld}.\exists \texttt{hasChld}.\mathit{female}$$

In $\mathcal{GF}$:

$$\forall x\ \texttt{grf-wth-gdtrs}(x) \to \exists y\ \texttt{hasChld}(x, y) \land (\exists z\ \texttt{hasChld}(y, z) \land \mathit{female}(z))$$

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Both $\mathcal{GF}$ and $\mathcal{FF}$ capture $\mathcal{ALC}$, e.g.: "Grandfathers with granddaughters"

$$\texttt{grf-wth-gdtrs} \sqsubseteq \exists \texttt{hasChld}.\exists \texttt{hasChld}.\textit{female}$$

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Both $\mathcal{GF}$ and $\mathcal{FF}$ capture $\mathcal{ALC}$, e.g.: "Grandfathers with granddaughters"

$$\texttt{grf-wth-gdtrs} \sqsubseteq \exists \texttt{hasChld}.\exists \texttt{hasChld}.\mathit{female}$$

In $\mathcal{FF}$ and $\mathcal{GF}$ (thus in $\mathcal{FGF}$):

$$\forall x_1\ \texttt{grf-wth-gdtrs}(x_1) \rightarrow \exists x_2\ \texttt{hasChld}(x_1, x_2) \wedge \exists x_3\ \texttt{hasChld}(x_2, x_3) \wedge \mathit{female}(x_3)$$

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Both $\mathcal{GF}$ and $\mathcal{FF}$ capture $\mathcal{ALC}$, e.g.: “Grandfathers with granddaughters”

$$\texttt{grf-wth-gdtrs} \sqsubseteq \exists\texttt{hasChld}.\exists\texttt{hasChld}.\mathit{female}$$

In $\mathcal{FF}$ and $\mathcal{GF}$ (thus in $\mathcal{FGF}$):

$$\forall x_1\ \texttt{grf-wth-gdtrs}(x_1) \rightarrow \exists x_2\ \texttt{hasChld}(x_1, x_2) \wedge \exists x_3\ \texttt{hasChld}(x_2, x_3) \wedge \mathit{female}(x_3)$$

Note that the Forward Guarded Fragment $\mathcal{FGF} := \mathcal{GF} \cap \mathcal{FF}$ also captures $\mathcal{ALC}$.

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Both $\mathcal{GF}$ and $\mathcal{FF}$ capture $\mathcal{ALC}$, e.g.: "Grandfathers with granddaughters"

$$\texttt{grf-wth-gdtrs} \sqsubseteq \exists \texttt{hasChld}.\exists \texttt{hasChld}.\mathit{female}$$

In $\mathcal{FF}$ and $\mathcal{GF}$ (thus in $\mathcal{FGF}$):

$$\forall x_1\ \texttt{grf-wth-gdtrs}(x_1) \to \exists x_2\ \texttt{hasChld}(x_1, x_2) \land \exists x_3\ \texttt{hasChld}(x_2, x_3) \land \mathit{female}(x_3)$$

Note that the Forward Guarded Fragment $\mathcal{FGF} := \mathcal{GF} \cap \mathcal{FF}$ also captures $\mathcal{ALC}$.

Nice remark: $\mathcal{FO}$ characterisation of formal languages

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Both $\mathcal{GF}$ and $\mathcal{FF}$ capture $\mathcal{ALC}$, e.g.: "Grandfathers with granddaughters"

$$\texttt{grf-wth-gdtrs} \sqsubseteq \exists \texttt{hasChld}.\exists \texttt{hasChld}.\mathit{female}$$

In $\mathcal{FF}$ and $\mathcal{GF}$ (thus in $\mathcal{FGF}$):

$$\forall x_1\, \texttt{grf-wth-gdtrs}(x_1) \rightarrow \exists x_2\, \texttt{hasChld}(x_1, x_2) \land \exists x_3\, \texttt{hasChld}(x_2, x_3) \land \mathit{female}(x_3)$$

Note that the Forward Guarded Fragment $\mathcal{FGF} := \mathcal{GF} \cap \mathcal{FF}$ also captures $\mathcal{ALC}$.

Nice remark: $\mathcal{FO}$ characterisation of formal languages

LTL corresponds to $\mathcal{FO}[\leq]$ over words, LTL[$\mathbf{XF}, \mathbf{XP}$] corresponds to $\mathcal{FO}^2[\leq]$

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Both $\mathcal{GF}$ and $\mathcal{FF}$ capture $\mathcal{ALC}$, e.g.: "Grandfathers with granddaughters"

$$\texttt{grf-wth-gdtrs} \sqsubseteq \exists\texttt{hasChld}.\exists\texttt{hasChld}.\mathit{female}$$

In $\mathcal{FF}$ and $\mathcal{GF}$ (thus in $\mathcal{FGF}$):

$$\forall x_1\, \texttt{grf-wth-gdtrs}(x_1) \to \exists x_2\, \texttt{hasChld}(x_1, x_2) \land \exists x_3\, \texttt{hasChld}(x_2, x_3) \land \mathit{female}(x_3)$$

Note that the Forward Guarded Fragment $\mathcal{FGF} := \mathcal{GF} \cap \mathcal{FF}$ also captures $\mathcal{ALC}$.

Nice remark: $\mathcal{FO}$ characterisation of formal languages

LTL corresponds to $\mathcal{FO}[\leq]$ over words, LTL$[\mathbf{XF}, \mathbf{XP}]$ corresponds to $\mathcal{FO}^2[\leq]$

Is there any logic equivalent to LTL$[\mathbf{F}]$ and LTL$[\mathbf{XF}]$ over words?

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Both $\mathcal{GF}$ and $\mathcal{FF}$ capture $\mathcal{ALC}$, e.g.: "Grandfathers with granddaughters"

$$\texttt{grf-wth-gdtrs} \sqsubseteq \exists \texttt{hasChld}.\exists \texttt{hasChld}.\mathit{female}$$

In $\mathcal{FF}$ and $\mathcal{GF}$ (thus in $\mathcal{FGF}$):

$$\forall x_1\, \texttt{grf-wth-gdtrs}(x_1) \to \exists x_2\, \texttt{hasChld}(x_1, x_2) \land \exists x_3\, \texttt{hasChld}(x_2, x_3) \land \mathit{female}(x_3)$$

Note that the Forward Guarded Fragment $\mathcal{FGF} := \mathcal{GF} \cap \mathcal{FF}$ also captures $\mathcal{ALC}$.

Nice remark: $\mathcal{FO}$ characterisation of formal languages

LTL corresponds to $\mathcal{FO}[\leq]$ over words, LTL[$\mathbf{XF}, \mathbf{XP}$] corresponds to $\mathcal{FO}^2[\leq]$

Is there any logic equivalent to LTL[$\mathbf{F}$] and LTL[$\mathbf{XF}$] over words?

Yes! $\mathcal{FGF}[\leq]$ and $\mathcal{FGF}[<]$ ☺

On intersection of $\mathcal{GF}$ [Andreka et al. 1998] and $\mathcal{FL}$ [Quine 1969]

Both $\mathcal{GF}$ and $\mathcal{FF}$ capture $\mathcal{ALC}$, e.g.: "Grandfathers with granddaughters"

$$\texttt{grf-wth-gdtrs} \sqsubseteq \exists \texttt{hasChld}.\exists \texttt{hasChld}.\mathit{female}$$

In $\mathcal{FF}$ and $\mathcal{GF}$ (thus in $\mathcal{FGF}$):

$$\forall x_1\, \texttt{grf-wth-gdtrs}(x_1) \to \exists x_2\, \texttt{hasChld}(x_1, x_2) \land \exists x_3\, \texttt{hasChld}(x_2, x_3) \land \mathit{female}(x_3)$$

Note that the Forward Guarded Fragment $\mathcal{FGF} := \mathcal{GF} \cap \mathcal{FF}$ also captures $\mathcal{ALC}$.

Nice remark: $\mathcal{FO}$ characterisation of formal languages

LTL corresponds to $\mathcal{FO}[\leq]$ over words, LTL[$\mathbf{XF}, \mathbf{XP}$] corresponds to $\mathcal{FO}^2[\leq]$

Is there any logic equivalent to LTL[$\mathbf{F}$] and LTL[$\mathbf{XF}$] over words?

Yes! $\mathcal{FGF}[\leq]$ and $\mathcal{FGF}[<]$ ☺

Theorem (TFAE for a formal language $\mathcal{L} \subseteq \Sigma^*$)

(a) $\mathcal{L}$ is definable in $\mathcal{FGF}[\leq]$, (b) is def. in LTL[$\mathbf{XF}$],

(c) is rec. by partially-ordered 1way DFA, (d) $\mathrm{M}(\mathcal{L})$ belongs to the variety $\mathbf{R}$

(e) $\mathcal{L}$ is a fin disj. union $A_0^* a_1 A_1^* \ldots a_k A_k^*$ with $a_i \in \Sigma$, $A_i \subseteq \Sigma$ and $a_i \notin A_{i-1}$.

The Forward Guarded Fragment $\mathcal{FGF}$ [B., JELIA 2021]: Our results

The Forward Guarded Fragment $\mathcal{FGF}$ [B., JELIA 2021]: Our results

- New, arguably elegant logic $\mathcal{FGF}$ over relational, equality-free signatures.

The Forward Guarded Fragment $\mathcal{FGF}$ [B., JELIA 2021]: Our results

- New, arguably elegant logic $\mathcal{FGF}$ over relational, equality-free signatures.
- $\mathcal{FGF}$ cannot express "bad guys": transitivity, self-loops, nominals and inverses.

The Forward Guarded Fragment $\mathcal{FGF}$ [B., JELIA 2021]: Our results

- New, arguably elegant logic $\mathcal{FGF}$ over relational, equality-free signatures.
- $\mathcal{FGF}$ cannot express "bad guys": transitivity, self-loops, nominals and inverses.

$\varphi_{\mathrm{tr}(R)} = \forall x_1 \forall x_2 \forall x_3 \; R(x_1, x_2) \wedge R(x_2, x_3) \rightarrow R(x_1, x_3).$

The Forward Guarded Fragment $\mathcal{FGF}$ [B., JELIA 2021]: Our results

- New, arguably elegant logic $\mathcal{FGF}$ over relational, equality-free signatures.
- $\mathcal{FGF}$ cannot express "bad guys": transitivity, self-loops, nominals and inverses.

$\varphi_{\mathrm{tr}(R)} = \forall x_1 \forall x_2 \forall x_3\ R(x_1, x_2) \wedge R(x_2, x_3) \rightarrow R(x_1, x_3).$

$\varphi_{\mathrm{loop}(R)}(x_1) = R(x_1, x_1).$

The Forward Guarded Fragment $\mathcal{FGF}$ [B., JELIA 2021]: Our results

- New, arguably elegant logic $\mathcal{FGF}$ over relational, equality-free signatures.
- $\mathcal{FGF}$ cannot express "bad guys": transitivity, self-loops, nominals and inverses.

$\varphi_{\mathrm{tr}(R)} = \forall x_1 \forall x_2 \forall x_3 \; R(x_1, x_2) \wedge R(x_2, x_3) \rightarrow R(x_1, x_3)$.

$\varphi_{\mathrm{loop}(R)}(x_1) = R(x_1, x_1)$.

$\varphi_{\mathrm{inv}(S)=R} := \forall x_1 x_2 S(x_1, x_2) \leftrightarrow R(x_2, x_1)$

The Forward Guarded Fragment $\mathcal{FGF}$ [B., JELIA 2021]: Our results

- New, arguably elegant logic $\mathcal{FGF}$ over relational, equality-free signatures.
- $\mathcal{FGF}$ cannot express "bad guys": transitivity, self-loops, nominals and inverses.

$$\varphi_{\mathrm{tr}(R)} = \forall x_1 \forall x_2 \forall x_3 \; R(x_1, x_2) \land R(x_2, x_3) \to R(x_1, x_3).$$

$$\varphi_{\mathrm{loop}(R)}(x_1) = R(x_1, x_1).$$

$$\varphi_{\mathrm{inv}(S)=R} := \forall x_1 x_2 S(x_1, x_2) \leftrightarrow R(x_2, x_1)$$

$$\varphi_{\mathrm{unique}(A)} := \forall x_1 x_2 \; \underbrace{A(x_1) \land A(x_2)}_{\text{not guarded!}} \to x_1 = x_2$$

The Forward Guarded Fragment $\mathcal{FGF}$ [B., JELIA 2021]: Our results

- New, arguably elegant logic $\mathcal{FGF}$ over relational, equality-free signatures.
- $\mathcal{FGF}$ cannot express "bad guys": transitivity, self-loops, nominals and inverses.

$\varphi_{\mathrm{tr}(R)} = \forall x_1 \forall x_2 \forall x_3 \; R(x_1, x_2) \wedge R(x_2, x_3) \rightarrow R(x_1, x_3)$.

$\varphi_{\mathrm{loop}(R)}(x_1) = R(x_1, x_1)$.

$\varphi_{\mathrm{inv}(S)=R} := \forall x_1 x_2 S(x_1, x_2) \leftrightarrow R(x_2, x_1)$

$\varphi_{\mathrm{unique}(A)} := \forall x_1 x_2 \; \underbrace{A(x_1) \wedge A(x_2)}_{\text{not guarded!}} \rightarrow x_1 = x_2$

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ are $\textsc{ExpTime}$-complete.

The Forward Guarded Fragment $\mathcal{FGF}$ [B., JELIA 2021]: Our results

- New, arguably elegant logic $\mathcal{FGF}$ over relational, equality-free signatures.
- $\mathcal{FGF}$ cannot express "bad guys": transitivity, self-loops, nominals and inverses.

$\varphi_{tr(R)} = \forall x_1 \forall x_2 \forall x_3\; R(x_1, x_2) \wedge R(x_2, x_3) \rightarrow R(x_1, x_3)$.

$\varphi_{loop(R)}(x_1) = R(x_1, x_1)$.

$\varphi_{inv(S)=R} := \forall x_1 x_2 S(x_1, x_2) \leftrightarrow R(x_2, x_1)$

$\varphi_{unique(A)} := \forall x_1 x_2\; \underbrace{A(x_1) \wedge A(x_2)}_{\text{not guarded!}} \rightarrow x_1 = x_2$

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ are ExpTime-complete.

Harvesting from the results of Grädel and Bárány et al:

The Forward Guarded Fragment $\mathcal{FGF}$ [B., JELIA 2021]: Our results

- New, arguably elegant logic $\mathcal{FGF}$ over relational, equality-free signatures.
- $\mathcal{FGF}$ cannot express "bad guys": transitivity, self-loops, nominals and inverses.

$$\varphi_{\mathrm{tr}(R)} = \forall x_1 \forall x_2 \forall x_3 \; R(x_1, x_2) \land R(x_2, x_3) \to R(x_1, x_3).$$

$$\varphi_{\mathrm{loop}(R)}(x_1) = R(x_1, x_1).$$

$$\varphi_{\mathrm{inv}(S)=R} := \forall x_1 x_2 S(x_1, x_2) \leftrightarrow R(x_2, x_1)$$

$$\varphi_{\mathrm{unique}(A)} := \forall x_1 x_2 \; \underbrace{A(x_1) \land A(x_2)}_{\text{not guarded!}} \to x_1 = x_2$$

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ are $\textsc{ExpTime}$-complete.

Harvesting from the results of Grädel and Bárány et al:

Corollary

Data complexity of KB SAT is NP-compl and coNP-compl for querying.

$\mathcal{FGF}$ has FMP and is finitely-controllable.

Main ingredients for SAT: Part I

Main ingredients for SAT: Part I

Definition (Forward type)

A (Σ, n)-*forward type* is a conjunction of atoms with n free-variables $\vec{x}_{1 \ldots n}$, which for every relational symbol $\mathrm{R} \in \Sigma$ of arity $\ell = \mathrm{ar}(\mathrm{R}) \leq n$ and every index $1 \leq i \leq n+1-\ell$ contains either $\mathrm{R}(\vec{x}_{i \ldots i+\ell-1})$ or $\neg\mathrm{R}(\vec{x}_{i \ldots i+\ell-1})$.

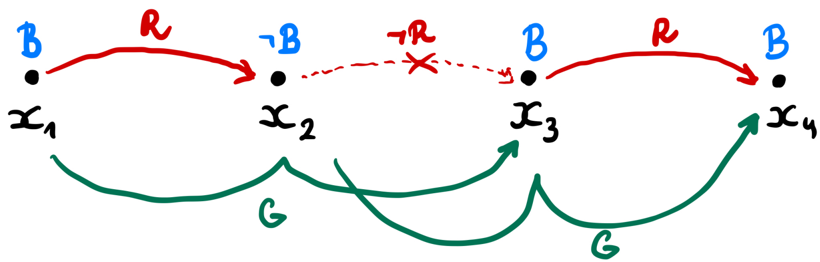
Main ingredients for SAT: Part I

Definition (Forward type)

A (Σ, n)-*forward type* is a conjunction of atoms with n free-variables $\vec{x}_{1\dots n}$, which for every relational symbol $\mathrm{R} \in \Sigma$ of arity $\ell = \mathrm{ar}(\mathrm{R}) \leq n$ and every index $1 \leq i \leq n+1-\ell$ contains either $\mathrm{R}(\vec{x}_{i\dots i+\ell-1})$ or $\neg\mathrm{R}(\vec{x}_{i\dots i+\ell-1})$.

Blue B^1, Red R^2, Green G^3

$(\{R, G, B\}, 4)$-forward tp

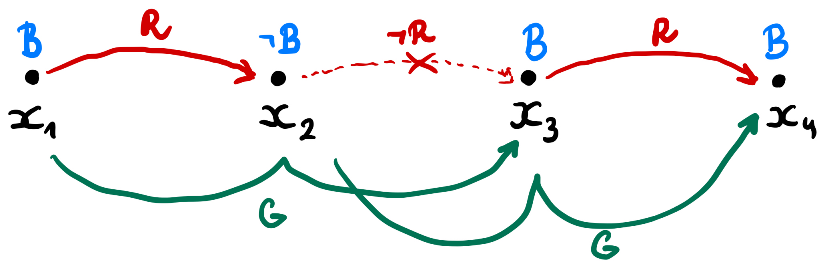


Definition (Forward type)

A (Σ, n)-*forward type* is a conjunction of atoms with n free-variables $\vec{x}_{1\dots n}$, which for every relational symbol $\mathrm{R} \in \Sigma$ of arity $\ell = \mathrm{ar}(\mathrm{R}) \leq n$ and every index $1 \leq i \leq n{+}1{-}\ell$ contains either $\mathrm{R}(\vec{x}_{i\dots i+\ell-1})$ or $\neg\mathrm{R}(\vec{x}_{i\dots i+\ell-1})$.

Blue B^1, Red R^2, Green G^3

$(\{R, G, B\}, 4)$-forward tp



Lemma

The number of different (Σ, n)-types is $\leq 2^{|\Sigma| \cdot n^2}$.

The number of conjuncts in each (Σ, n)-type is $\leq |\Sigma| \cdot n$

Main ingredients for SAT: Part II

Main ingredients for SAT: Part II

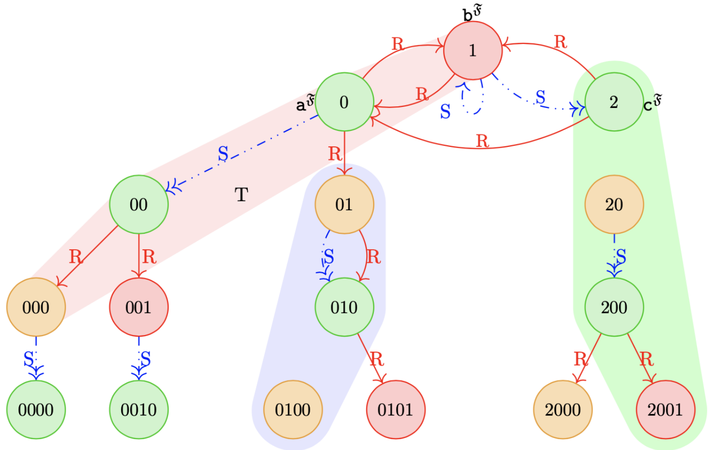
Definition (Higher-arity forests (HAFs))

There are forests in which (higher-arity) edges link roots in arbitrary way but other elements are connected in the level-by-level order.

Main ingredients for SAT: Part II

Definition (Higher-arity forests (HAFs))

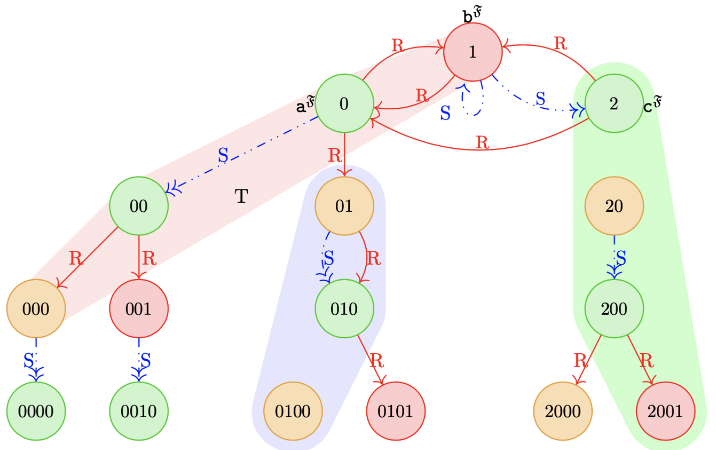
There are forests in which (higher-arity) edges link roots in arbitrary way but other elements are connected in the level-by-level order.



Main ingredients for SAT: Part II

Definition (Higher-arity forests (HAFs))

There are forests in which (higher-arity) edges link roots in arbitrary way but other elements are connected in the level-by-level order.



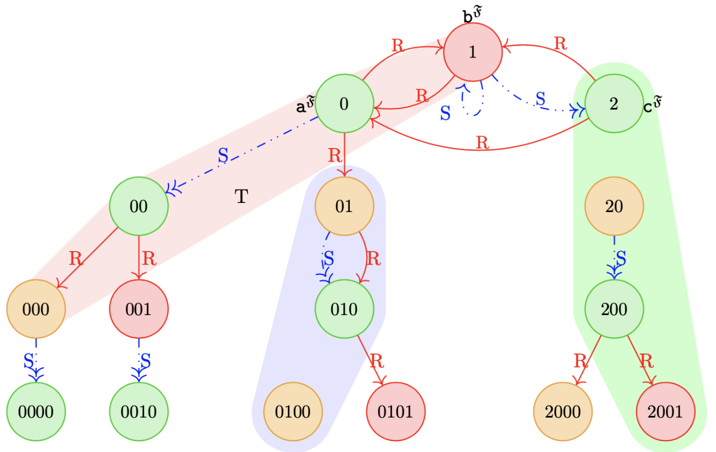
Lemma

Every satisfiable $\mathcal{FGF}$ knowledge base has a HAF (counter)model.

Main ingredients for SAT: Part II

Definition (Higher-arity forests (HAFs))

There are forests in which (higher-arity) edges link roots in arbitrary way but other elements are connected in the level-by-level order.



Lemma

Every satisfiable $\mathcal{FGF}$ knowledge base has a HAF (counter)model.

Proof

via suitable notion of HAF-unravelling, similar to [BBR, ECAI'20]

Main ingredients for SAT: Part III (last)

Main ingredients for SAT: Part III (last)

1. Take a satisfiable $\mathcal{FGF}$ knowledge base $\mathcal{K}$ and any of its models $\mathfrak{A}$.

Main ingredients for SAT: Part III (last)

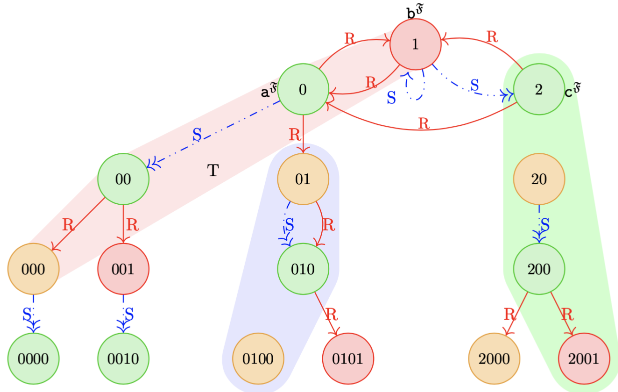
1. Take a satisfiable $\mathcal{FGF}$ knowledge base $\mathcal{K}$ and any of its models $\mathfrak{A}$.

2. Transform $\mathfrak{A}$ into a HAF model $\mathfrak{F}$ of $\mathcal{K}$.

Main ingredients for SAT: Part III (last)

1. Take a satisfiable $\mathcal{FGF}$ knowledge base $\mathcal{K}$ and any of its models $\mathfrak{A}$.

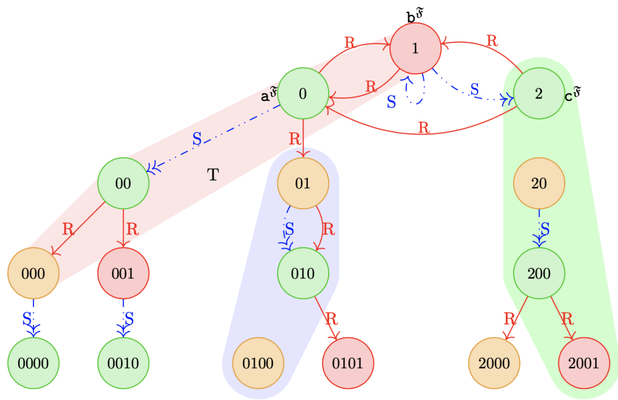
2. Transform $\mathfrak{A}$ into a HAF model $\mathfrak{F}$ of $\mathcal{K}$.



Main ingredients for SAT: Part III (last)

1. Take a satisfiable $\mathcal{FGF}$ knowledge base $\mathcal{K}$ and any of its models $\mathfrak{A}$.

2. Transform $\mathfrak{A}$ into a HAF model $\mathfrak{F}$ of $\mathcal{K}$.

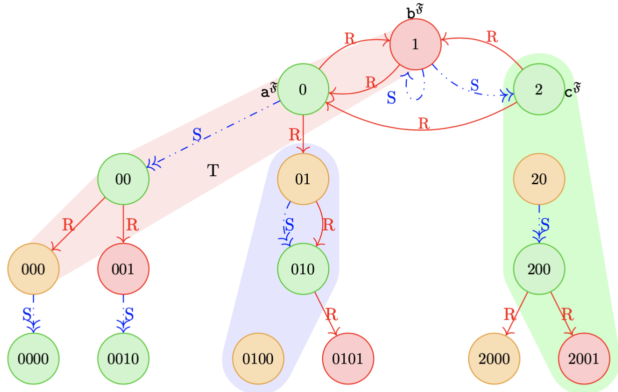


3. Make $\mathfrak{F}$ root-sparse, i.e. $\mathfrak{F}\restriction_{F\cap\mathbb{N}}$ should have $\leq$ *poly*$(\mathcal{K})$ tuples in relations.

Main ingredients for SAT: Part III (last)

1. Take a satisfiable $\mathcal{FGF}$ knowledge base $\mathcal{K}$ and any of its models $\mathfrak{A}$.

2. Transform $\mathfrak{A}$ into a HAF model $\mathfrak{F}$ of $\mathcal{K}$.



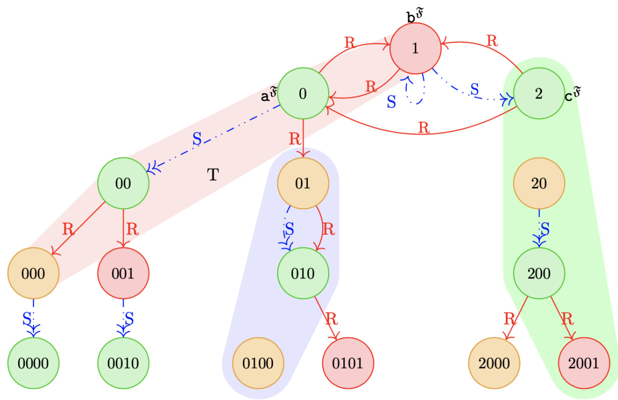
3. Make $\mathfrak{F}$ root-sparse, i.e. $\mathfrak{F}\restriction_{F\cap\mathbb{N}}$ should have $\leq$ *poly*$(\mathcal{K})$ tuples in relations.

4. Do some pruning to establish that degree of each node is $\leq$ *poly*$(\mathcal{K})$.

Main ingredients for SAT: Part III (last)

1. Take a satisfiable $\mathcal{FGF}$ knowledge base $\mathcal{K}$ and any of its models $\mathfrak{A}$.

2. Transform $\mathfrak{A}$ into a HAF model $\mathfrak{F}$ of $\mathcal{K}$.



3. Make $\mathfrak{F}$ root-sparse, i.e. $\mathfrak{F}\restriction_{F \cap \mathbb{N}}$ should have $\leq$ *poly*$(\mathcal{K})$ tuples in relations.

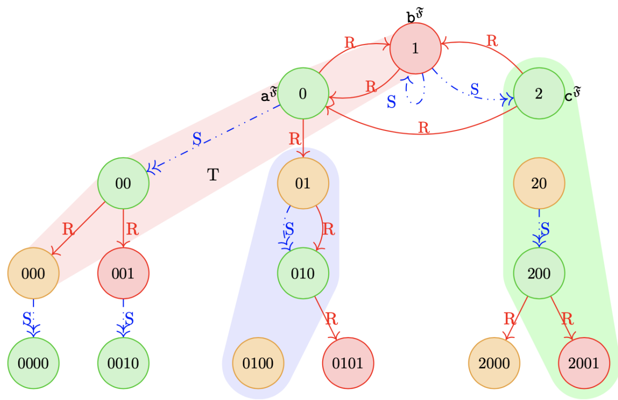
4. Do some pruning to establish that degree of each node is $\leq$ *poly*$(\mathcal{K})$.

5. The "relevant" part of $\mathfrak{F}$ is of depth $\leq$ *poly*(number of types) $=$ *exp*$(\mathcal{K})$.

Main ingredients for SAT: Part III (last)

1. Take a satisfiable $\mathcal{FGF}$ knowledge base $\mathcal{K}$ and any of its models $\mathfrak{A}$.

2. Transform $\mathfrak{A}$ into a HAF model $\mathfrak{F}$ of $\mathcal{K}$.



3. Make $\mathfrak{F}$ root-sparse, i.e. $\mathfrak{F}\restriction_{F\cap\mathbb{N}}$ should have $\leq poly(\mathcal{K})$ tuples in relations.

4. Do some pruning to establish that degree of each node is $\leq poly(\mathcal{K})$.

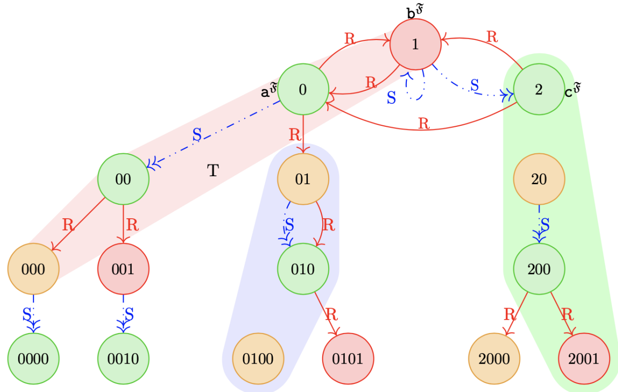
5. The "relevant" part of $\mathfrak{F}$ is of depth $\leq poly(\text{number of types}) = exp(\mathcal{K})$.

6. Use APSPACE tableaux-like procedure to construct the relevant part of $\mathfrak{F}$.

Main ingredients for SAT: Part III (last)

1. Take a satisfiable $\mathcal{FGF}$ knowledge base $\mathcal{K}$ and any of its models $\mathfrak{A}$.

2. Transform $\mathfrak{A}$ into a HAF model $\mathfrak{F}$ of $\mathcal{K}$.



3. Make $\mathfrak{F}$ root-sparse, i.e. $\mathfrak{F}\restriction_{F\cap\mathbb{N}}$ should have $\leq \mathit{poly}(\mathcal{K})$ tuples in relations.

4. Do some pruning to establish that degree of each node is $\leq \mathit{poly}(\mathcal{K})$.

5. The "relevant" part of $\mathfrak{F}$ is of depth $\leq \mathit{poly}(\text{number of types}) = \mathit{exp}(\mathcal{K})$.

6. Use $\textsc{APSpace}$ tableaux-like procedure to construct the relevant part of $\mathfrak{F}$.

Theorem (B., JELIA 2021)

Knowledge-base SAT for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete.

Main ingredients for Query entailment: Part I (intro)

Main ingredients for Query entailment: Part I (intro)

Recap: Conjunctive query is a conjunction of positive atoms.

Def: $\mathcal{K} \models q$ iff for all models $\mathfrak{A}$ of $\mathcal{K}$ we have $\mathfrak{A} \models q$ (query q matches $\mathfrak{A}$)

If $\mathfrak{A} \models \mathcal{K}$ but $\mathfrak{A} \not\models q$ we call $\mathfrak{A}$ a countermodel for $(\mathcal{K}, q)$.

Main ingredients for Query entailment: Part I (intro)

Recap: Conjunctive query is a conjunction of positive atoms.

Def: $\mathcal{K} \models q$ iff for all models $\mathfrak{A}$ of $\mathcal{K}$ we have $\mathfrak{A} \models q$ (query q matches $\mathfrak{A}$)

If $\mathfrak{A} \models \mathcal{K}$ but $\mathfrak{A} \not\models q$ we call $\mathfrak{A}$ a countermodel for $(\mathcal{K}, q)$.

Lemma

If there is countermodel for $(\mathcal{K}, q)$ then there is also a HAF countermodel.

Main ingredients for Query entailment: Part I (intro)

Recap: Conjunctive query is a conjunction of positive atoms.

Def: $\mathcal{K} \models q$ iff for all models $\mathfrak{A}$ of $\mathcal{K}$ we have $\mathfrak{A} \models q$ (query q matches $\mathfrak{A}$)

If $\mathfrak{A} \models \mathcal{K}$ but $\mathfrak{A} \not\models q$ we call $\mathfrak{A}$ a countermodel for $(\mathcal{K}, q)$.

Lemma

If there is countermodel for $(\mathcal{K}, q)$ then there is also a HAF countermodel.

Caveat: W.l.o.g. we assume that queries are preffix and suffix closed, e.g.

$$\text{if } \mathrm{U}(x_1, x_2, x_3, x_4) \in q \text{ then } \mathrm{U}_3(x_1, x_2, x_3) \in q$$

Main ingredients for Query entailment: Part I (intro)

Recap: Conjunctive query is a conjunction of positive atoms.

Def: $\mathcal{K} \models q$ iff for all models $\mathfrak{A}$ of $\mathcal{K}$ we have $\mathfrak{A} \models q$ (query q matches $\mathfrak{A}$)

If $\mathfrak{A} \models \mathcal{K}$ but $\mathfrak{A} \not\models q$ we call $\mathfrak{A}$ a countermodel for $(\mathcal{K}, q)$.

Lemma

If there is countermodel for $(\mathcal{K}, q)$ then there is also a HAF countermodel.

Caveat: W.l.o.g. we assume that queries are preffix and suffix closed, e.g.

if $\mathrm{U}(x_1, x_2, x_3, x_4) \in q$ then $\mathrm{U}_3(x_1, x_2, x_3) \in q$

The first important step: how to query with HAF-shaped queries?

Main ingredients for Query entailment: Part I (intro)

Recap: Conjunctive query is a conjunction of positive atoms.

Def: $\mathcal{K} \models q$ iff for all models $\mathfrak{A}$ of $\mathcal{K}$ we have $\mathfrak{A} \models q$ (query q matches $\mathfrak{A}$)

If $\mathfrak{A} \models \mathcal{K}$ but $\mathfrak{A} \not\models q$ we call $\mathfrak{A}$ a countermodel for $(\mathcal{K}, q)$.

Lemma

If there is countermodel for $(\mathcal{K}, q)$ then there is also a HAF countermodel.

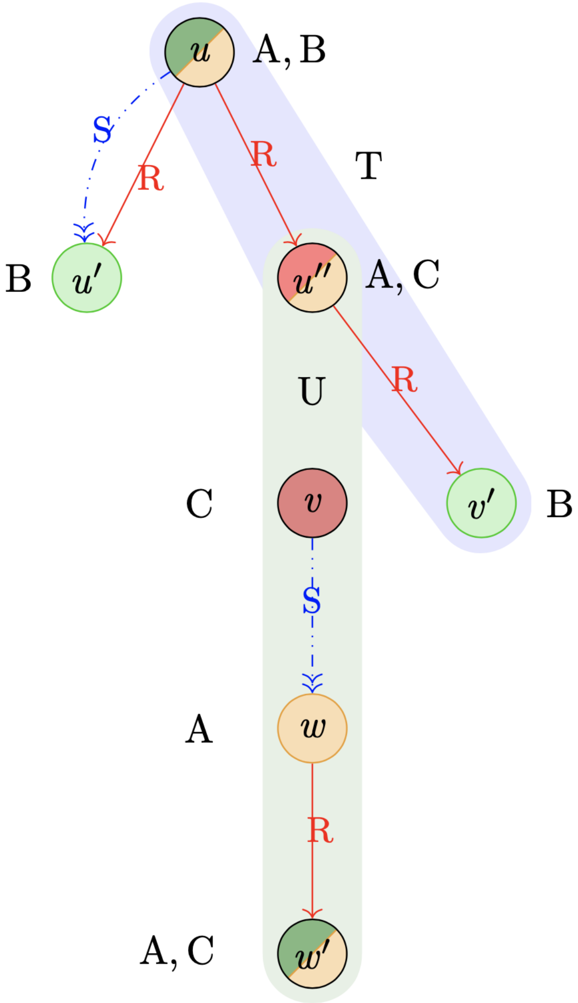
Caveat: W.l.o.g. we assume that queries are preffix and suffix closed, e.g.

if $\mathrm{U}(x_1, x_2, x_3, x_4) \in q$ then $\mathrm{U}_3(x_1, x_2, x_3) \in q$

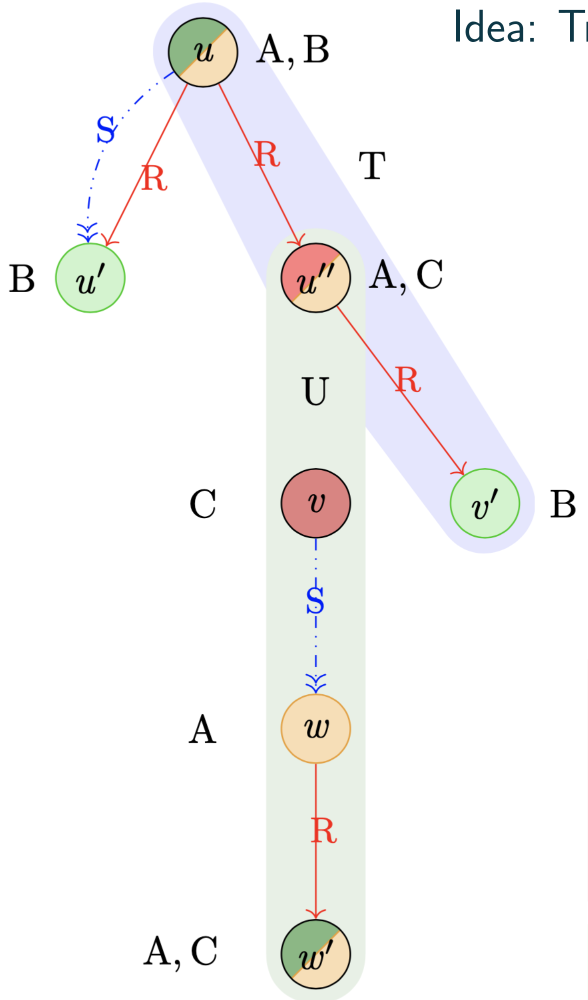
The first important step: how to query with HAF-shaped queries?

Quite technical generalisation of the rolling-up technique of transforming tree-shaped matches into concepts.

Main ingredients for Query entailment: Part II (rolling-up)

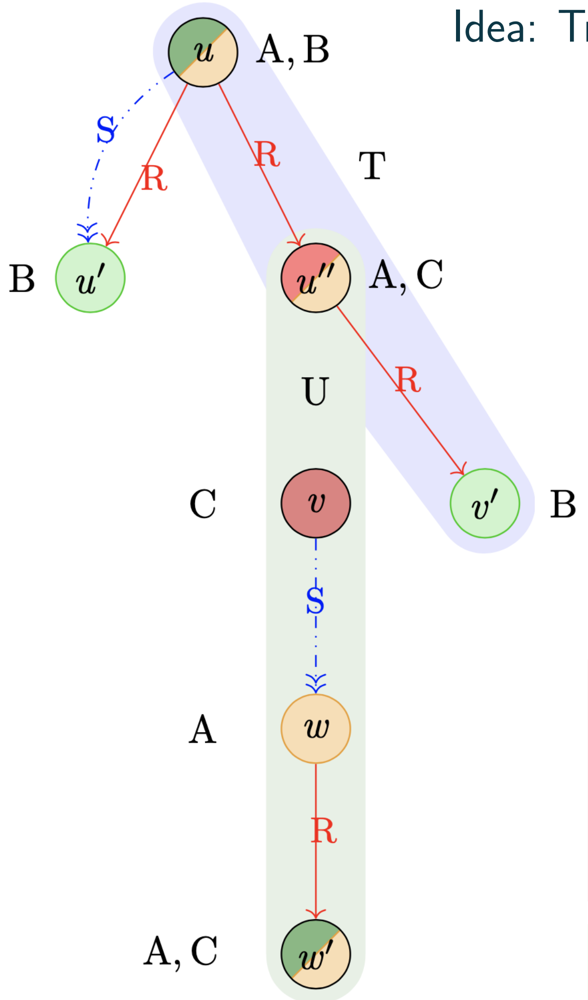


Main ingredients for Query entailment: Part II (rolling-up)



Idea: Traverse top-down and construct predicates $\mathrm{Subt}^{\star}_{q}(\star)$.

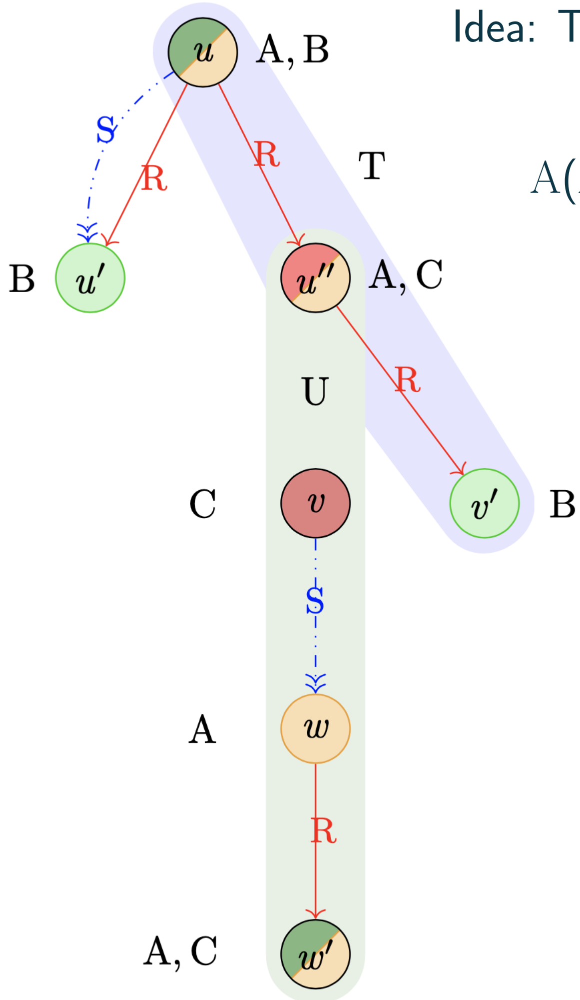
Main ingredients for Query entailment: Part II (rolling-up)



Idea: Traverse top-down and construct predicates $\mathrm{Subt}_q^{\star}(\star)$.

$$\mathrm{Match}_q(x_1) := \mathrm{Subt}_q^{u}(x_1) :=$$

Main ingredients for Query entailment: Part II (rolling-up)

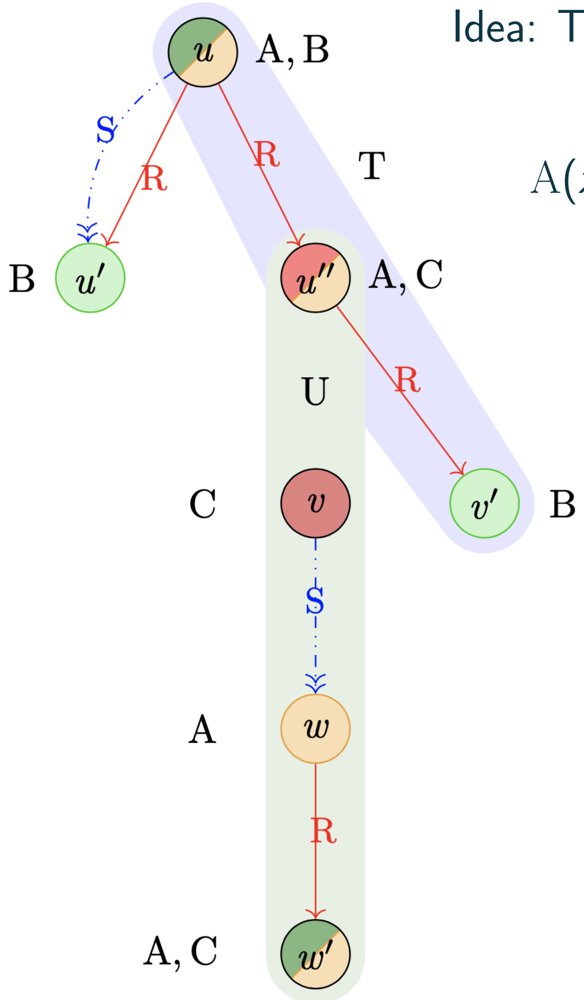


Idea: Traverse top-down and construct predicates $\mathrm{Subt}_q^{\star}(\star)$.

$$\mathrm{Match}_q(x_1) := \mathrm{Subt}_q^{u}(x_1) :=$$

$$\mathrm{A}(x_1) \wedge \mathrm{B}(x_1) \wedge \exists x_2 \mathrm{Subt}_q^{uu'}(x_1, x_2) \wedge \exists x_2 \mathrm{Subt}_q^{uu''}(x_1, x_2)$$

Main ingredients for Query entailment: Part II (rolling-up)



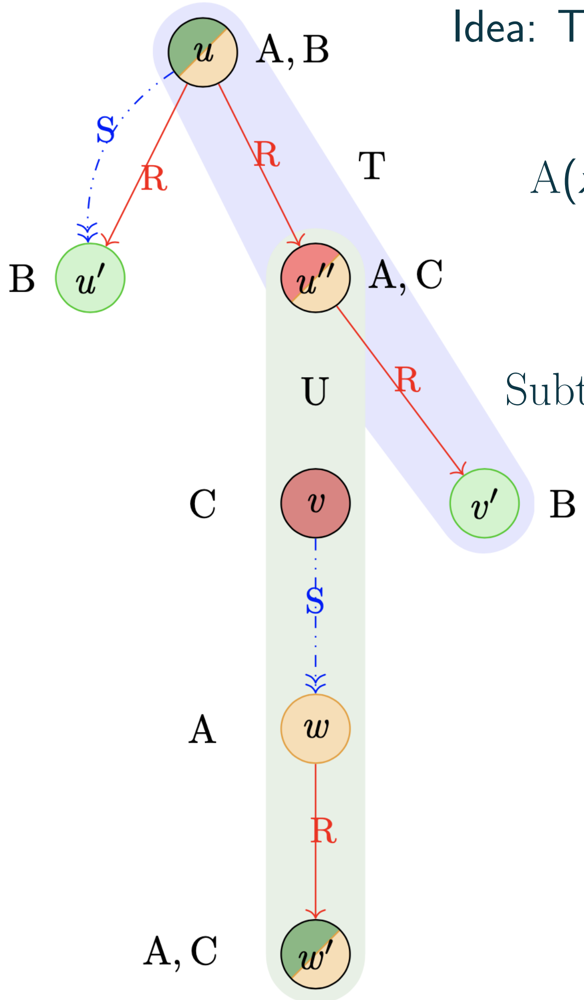
Idea: Traverse top-down and construct predicates $\mathrm{Subt}_q^{\star}(\star)$.

$$\mathrm{Match}_q(x_1) := \mathrm{Subt}_q^{u}(x_1) :=$$

$$\mathrm{A}(x_1) \wedge \mathrm{B}(x_1) \wedge \exists x_2 \mathrm{Subt}_q^{uu'}(x_1, x_2) \wedge \exists x_2 \mathrm{Subt}_q^{uu''}(x_1, x_2)$$

$$\mathrm{Subt}_q^{uu'}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{S}(x_1, x_2) \wedge \mathrm{B}(x_2)$$

Main ingredients for Query entailment: Part II (rolling-up)



Idea: Traverse top-down and construct predicates $\mathrm{Subt}_q^{\star}(\star)$.

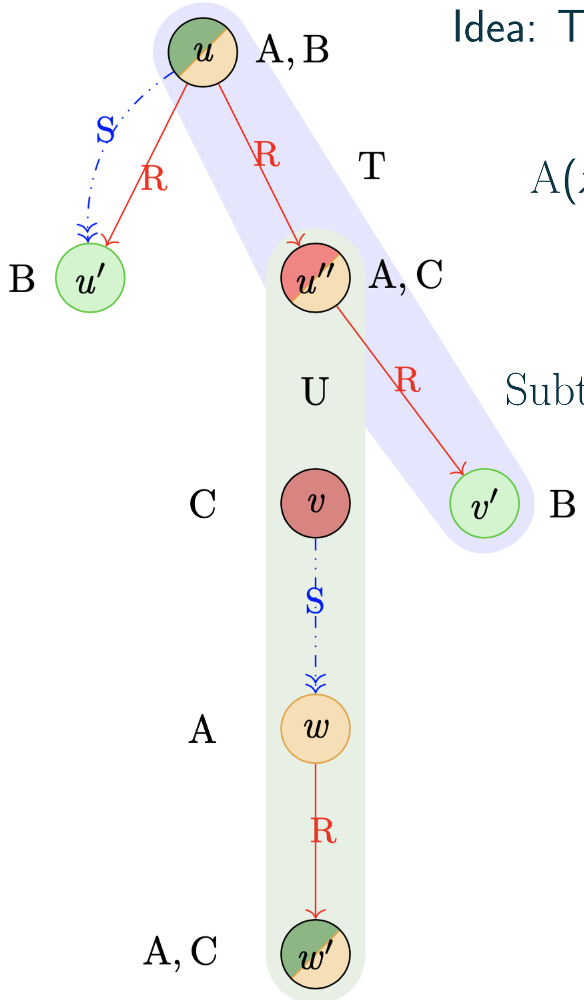
$$\mathrm{Match}_q(x_1) := \mathrm{Subt}_q^{u}(x_1) :=$$

$$\mathrm{A}(x_1) \wedge \mathrm{B}(x_1) \wedge \exists x_2 \mathrm{Subt}_q^{uu'}(x_1, x_2) \wedge \exists x_2 \mathrm{Subt}_q^{uu''}(x_1, x_2)$$

$$\mathrm{Subt}_q^{uu'}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{S}(x_1, x_2) \wedge \mathrm{B}(x_2)$$

$$\mathrm{Subt}_q^{uu''}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{T}_2(x_1, x_2) \wedge \mathrm{A}(x_2) \wedge \mathrm{C}(x_2)$$

Main ingredients for Query entailment: Part II (rolling-up)



Idea: Traverse top-down and construct predicates $\mathrm{Subt}_q^{\star}(\star)$.

$$\mathrm{Match}_q(x_1) := \mathrm{Subt}_q^{u}(x_1) :=$$

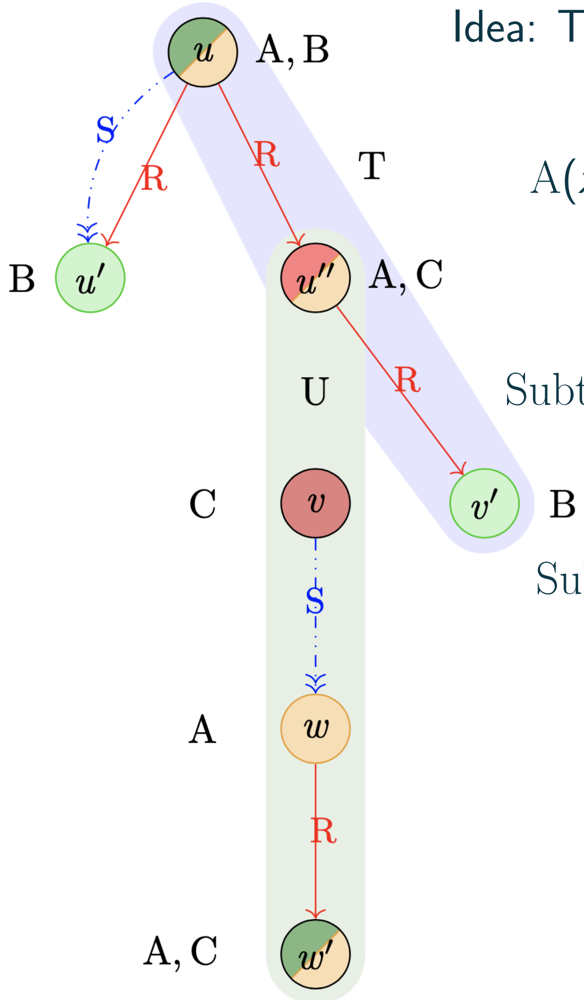
$$\mathrm{A}(x_1) \wedge \mathrm{B}(x_1) \wedge \exists x_2 \mathrm{Subt}_q^{uu'}(x_1, x_2) \wedge \exists x_2 \mathrm{Subt}_q^{uu''}(x_1, x_2)$$

$$\mathrm{Subt}_q^{uu'}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{S}(x_1, x_2) \wedge \mathrm{B}(x_2)$$

$$\mathrm{Subt}_q^{uu''}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{T}_2(x_1, x_2) \wedge \mathrm{A}(x_2) \wedge \mathrm{C}(x_2)$$

$$\wedge \exists x_3\, \mathrm{Subt}_q^{uu''v'}(x_1, x_2, x_3) \wedge \exists x_3\, \mathrm{Subt}_q^{u''v}(x_2, x_3)$$

Main ingredients for Query entailment: Part II (rolling-up)



Idea: Traverse top-down and construct predicates $\mathrm{Subt}_q^{\star}(\star)$.

$$\mathrm{Match}_q(x_1) := \mathrm{Subt}_q^{u}(x_1) :=$$

$$\mathrm{A}(x_1) \wedge \mathrm{B}(x_1) \wedge \exists x_2 \mathrm{Subt}_q^{uu'}(x_1, x_2) \wedge \exists x_2 \mathrm{Subt}_q^{uu''}(x_1, x_2)$$

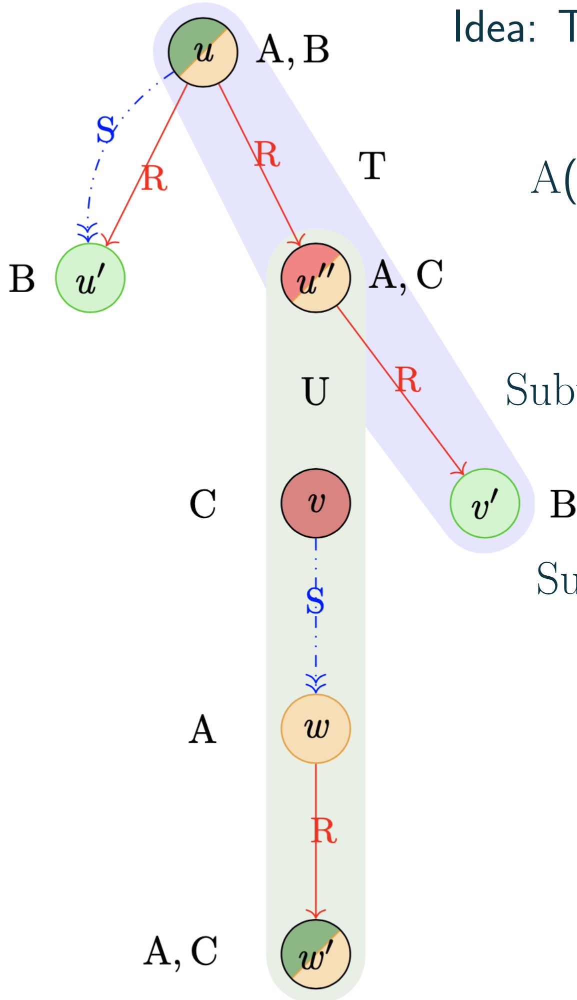
$$\mathrm{Subt}_q^{uu'}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{S}(x_1, x_2) \wedge \mathrm{B}(x_2)$$

$$\mathrm{Subt}_q^{uu''}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{T}_2(x_1, x_2) \wedge \mathrm{A}(x_2) \wedge \mathrm{C}(x_2)$$

$$\wedge \exists x_3\ \mathrm{Subt}_q^{uu''v'}(x_1, x_2, x_3) \wedge \exists x_3\ \mathrm{Subt}_q^{u''v}(x_2, x_3)$$

$$\mathrm{Subt}_q^{uu''v'}(x_1, x_2, x_3) := \mathrm{T}(x_1, x_2, x_3) \wedge \mathrm{B}(x_3) \wedge \mathrm{R}(x_2, x_3)$$

Main ingredients for Query entailment: Part II (rolling-up)



Idea: Traverse top-down and construct predicates $\mathrm{Subt}_q^{\star}(\star)$.

$$\mathrm{Match}_q(x_1) := \mathrm{Subt}_q^{u}(x_1) :=$$
$$\mathrm{A}(x_1) \wedge \mathrm{B}(x_1) \wedge \exists x_2 \mathrm{Subt}_q^{uu'}(x_1, x_2) \wedge \exists x_2 \mathrm{Subt}_q^{uu''}(x_1, x_2)$$

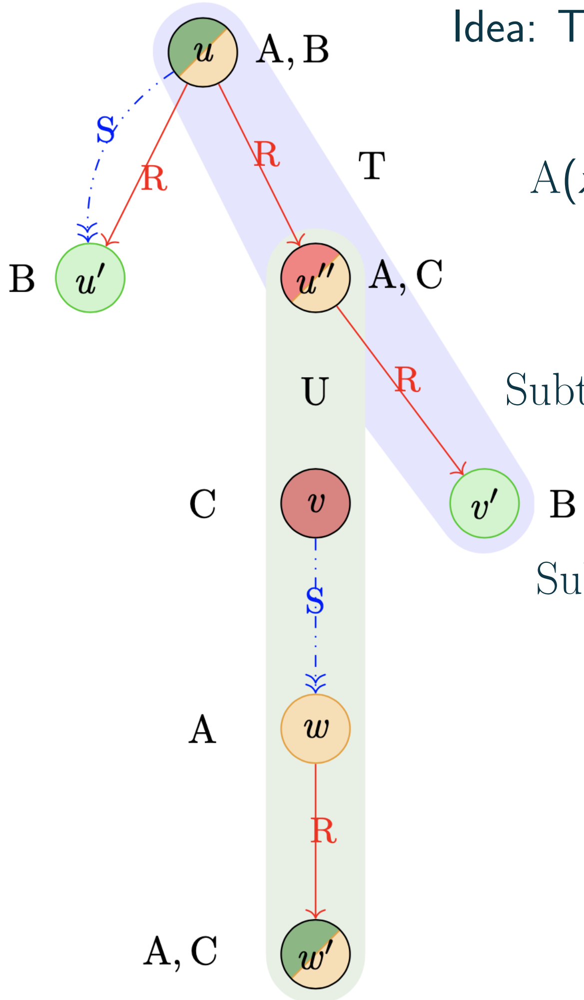
$$\mathrm{Subt}_q^{uu'}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{S}(x_1, x_2) \wedge \mathrm{B}(x_2)$$

$$\mathrm{Subt}_q^{uu''}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{T}_2(x_1, x_2) \wedge \mathrm{A}(x_2) \wedge \mathrm{C}(x_2)$$
$$\wedge \exists x_3 \, \mathrm{Subt}_q^{uu''v'}(x_1, x_2, x_3) \wedge \exists x_3 \, \mathrm{Subt}_q^{u''v}(x_2, x_3)$$

$$\mathrm{Subt}_q^{uu''v'}(x_1, x_2, x_3) := \mathrm{T}(x_1, x_2, x_3) \wedge \mathrm{B}(x_3) \wedge \mathrm{R}(x_2, x_3)$$

For any HAF-shaped CQ one can polytime compute the definition of $\mathrm{Match}_q(x_{root})$ with a meaning that $\mathrm{Match}_q^{\mathfrak{A}} \neq \emptyset$ iff $\mathfrak{A} \models q$.

Main ingredients for Query entailment: Part II (rolling-up)



Idea: Traverse top-down and construct predicates $\mathrm{Subt}_q^{\star}(\star)$.

$$\mathrm{Match}_q(x_1) := \mathrm{Subt}_q^{u}(x_1) :=$$

$$\mathrm{A}(x_1) \wedge \mathrm{B}(x_1) \wedge \exists x_2 \mathrm{Subt}_q^{uu'}(x_1, x_2) \wedge \exists x_2 \mathrm{Subt}_q^{uu''}(x_1, x_2)$$

$$\mathrm{Subt}_q^{uu'}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{S}(x_1, x_2) \wedge \mathrm{B}(x_2)$$

$$\mathrm{Subt}_q^{uu''}(x_1, x_2) := \mathrm{R}(x_1, x_2) \wedge \mathrm{T}_2(x_1, x_2) \wedge \mathrm{A}(x_2) \wedge \mathrm{C}(x_2)$$

$$\wedge \exists x_3 \, \mathrm{Subt}_q^{uu''v'}(x_1, x_2, x_3) \wedge \exists x_3 \, \mathrm{Subt}_q^{u''v}(x_2, x_3)$$

$$\mathrm{Subt}_q^{uu''v'}(x_1, x_2, x_3) := \mathrm{T}(x_1, x_2, x_3) \wedge \mathrm{B}(x_3) \wedge \mathrm{R}(x_2, x_3)$$

For any HAF-shaped CQ one can polytime compute the definition of $\mathrm{Match}_q(x_{root})$ with a meaning that $\mathrm{Match}_q^{\mathfrak{A}} \neq \emptyset$ iff $\mathfrak{A} \models q$.

$\mathcal{K} \not\models q_{haf}$ iff $\mathcal{K} \cup \{\forall x_1 \, \neg \mathrm{Match}_{q_{haf}}(x_1)\}$ is SAT.

Main ingredients for Querying: Part III (beyond HAF-shaped CQs)

Main ingredients for Querying: Part III (beyond HAF-shaped CQs)

To go beyond HAF-shaped CQs we need an auxiliary notion of a splitting.

Main ingredients for Querying: Part III (beyond HAF-shaped CQs)

To go beyond HAF-shaped CQs we need an auxiliary notion of a splitting.

Intuitively it mimics a query match by partitioning variables into three sets:

Main ingredients for Querying: Part III (beyond HAF-shaped CQs)

To go beyond HAF-shaped CQs we need an auxiliary notion of a splitting.

Intuitively it mimics a query match by partitioning variables into three sets:

(a) roots,

Main ingredients for Querying: Part III (beyond HAF-shaped CQs)

To go beyond HAF-shaped CQs we need an auxiliary notion of a splitting.

Intuitively it mimics a query match by partitioning variables into three sets:

(a) roots, (b) HAFs dangling from roots, and

Main ingredients for Querying: Part III (beyond HAF-shaped CQs)

To go beyond HAF-shaped CQs we need an auxiliary notion of a splitting.

Intuitively it mimics a query match by partitioning variables into three sets:

(a) roots, (b) HAFs dangling from roots, and (c) HAFs lying far from roots.

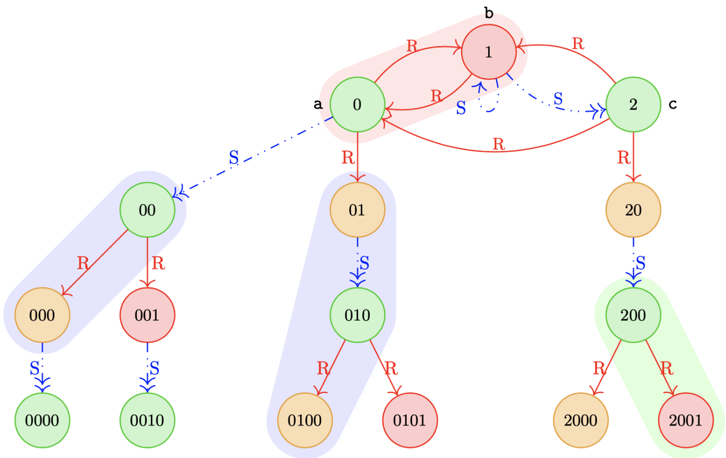
Main ingredients for Querying: Part III (beyond HAF-shaped CQs)

To go beyond HAF-shaped CQs we need an auxiliary notion of a splitting.

Intuitively it mimics a query match by partitioning variables into three sets:

(a) roots, (b) HAFs dangling from roots, and (c) HAFs lying far from roots.

$$q = (\mathrm{A}(x_0) \wedge \mathrm{R}(x_0, x_1) \wedge \mathrm{R}(x_1, x_0) \wedge \mathrm{B}(x_1)) \wedge (\mathrm{S}(x_0, x_{00}) \wedge \mathrm{R}(x_{00}, x_{000})) \\ \wedge (\mathrm{R}(x_0, x_{01}) \wedge \mathrm{S}(x_{01}, x_{010}) \wedge \mathrm{R}(x_{010}, x_{0100})) \wedge (\mathrm{A}(x_{200}) \wedge \mathrm{R}(x_{200}, x_{2001}) \wedge \mathrm{B}(x_{2001})) \,.$$



$$\mathrm{Roots} = \{\, x_0, x_1 \,\}$$

$$\mathrm{SubTree}_1 = \{\, x_{00}, x_{000} \,\}$$

$$\mathrm{SubTree}_2 = \{\, x_{01}, x_{010}, x_{0100} \,\}$$

$$\mathrm{Trees} = \{\, x_{200}, x_{2001} \,\}$$

$$\mathrm{name}(x_0) = \mathtt{a}, \mathrm{name}(x_1) = \mathtt{b}$$

$$\text{root-}\mathit{of}(1) = x_0, \text{root-}\mathit{of}(2) = x_0$$

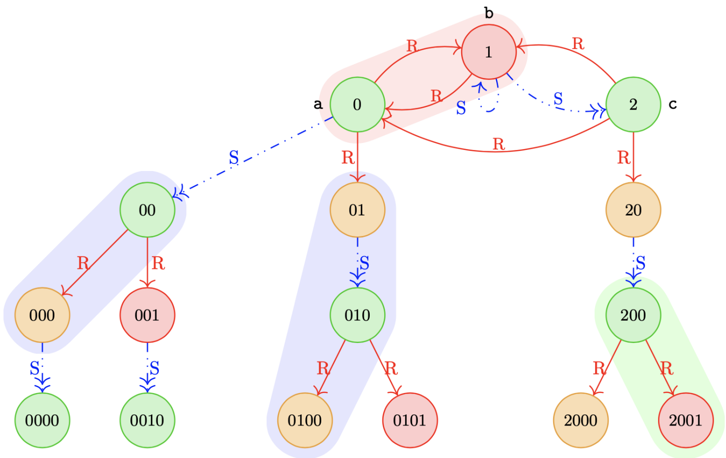
Main ingredients for Querying: Part III (beyond HAF-shaped CQs)

To go beyond HAF-shaped CQs we need an auxiliary notion of a splitting.

Intuitively it mimics a query match by partitioning variables into three sets:

(a) roots, (b) HAFs dangling from roots, and (c) HAFs lying far from roots.

$$q = (\mathrm{A}(x_0) \land \mathrm{R}(x_0, x_1) \land \mathrm{R}(x_1, x_0) \land \mathrm{B}(x_1)) \land (\mathrm{S}(x_0, x_{00}) \land \mathrm{R}(x_{00}, x_{000}))$$
$$\land (\mathrm{R}(x_0, x_{01}) \land \mathrm{S}(x_{01}, x_{010}) \land \mathrm{R}(x_{010}, x_{0100})) \land (\mathrm{A}(x_{200}) \land \mathrm{R}(x_{200}, x_{2001}) \land \mathrm{B}(x_{2001}))\,.$$



$$\mathrm{Roots} = \{\, x_0, x_1 \,\}$$

$$\mathrm{SubTree}_1 = \{\, x_{00}, x_{000} \,\}$$

$$\mathrm{SubTree}_2 = \{\, x_{01}, x_{010}, x_{0100} \,\}$$

$$\mathrm{Trees} = \{\, x_{200}, x_{2001} \,\}$$

$$\mathrm{name}(x_0) = \mathtt{a}, \mathrm{name}(x_1) = \mathtt{b}$$

$$\text{root-}\mathit{of}(1) = x_0, \text{root-}\mathit{of}(2) = x_0$$

With every splitting Π of q we associate a spoiler an $\mathcal{FGF}$-kb $\mathcal{K}_{\Pi}^{\lightning}$.

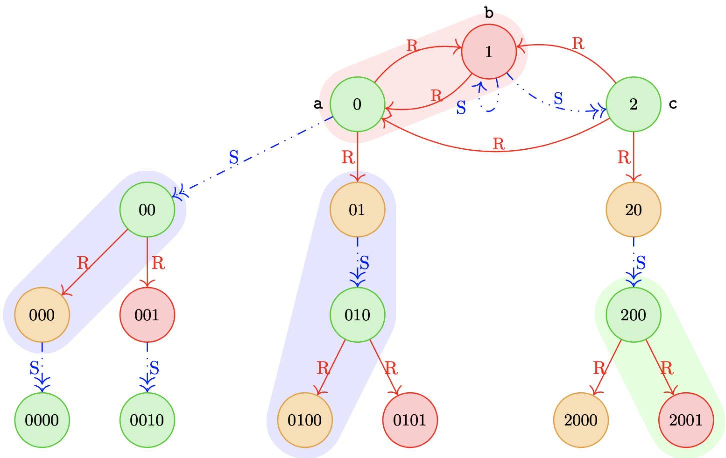
Main ingredients for Querying: Part III (beyond HAF-shaped CQs)

To go beyond HAF-shaped CQs we need an auxiliary notion of a splitting.

Intuitively it mimics a query match by partitioning variables into three sets:

(a) roots, (b) HAFs dangling from roots, and (c) HAFs lying far from roots.

$$q = (\mathrm{A}(x_0) \land \mathrm{R}(x_0, x_1) \land \mathrm{R}(x_1, x_0) \land \mathrm{B}(x_1)) \land (\mathrm{S}(x_0, x_{00}) \land \mathrm{R}(x_{00}, x_{000})) \\ \land (\mathrm{R}(x_0, x_{01}) \land \mathrm{S}(x_{01}, x_{010}) \land \mathrm{R}(x_{010}, x_{0100})) \land (\mathrm{A}(x_{200}) \land \mathrm{R}(x_{200}, x_{2001}) \land \mathrm{B}(x_{2001}))\,.$$



$$\mathrm{Roots} = \{\, x_0, x_1 \,\}$$

$$\mathrm{SubTree}_1 = \{\, x_{00}, x_{000} \,\}$$

$$\mathrm{SubTree}_2 = \{\, x_{01}, x_{010}, x_{0100} \,\}$$

$$\mathrm{Trees} = \{\, x_{200}, x_{2001} \,\}$$

$$\mathrm{name}(x_0) = \mathtt{a}, \mathrm{name}(x_1) = \mathtt{b}$$

$$\text{root-}\mathit{of}(1) = x_0, \text{root-}\mathit{of}(2) = x_0$$

With every splitting Π of q we associate a spoiler an $\mathcal{FGF}$-kb $\mathcal{K}_{\Pi}^{\lightning}$.

Idea: if $\mathcal{K} \cup \mathcal{K}_{\Pi}^{\lightning}$ then there is no matches of q splitting like Π.

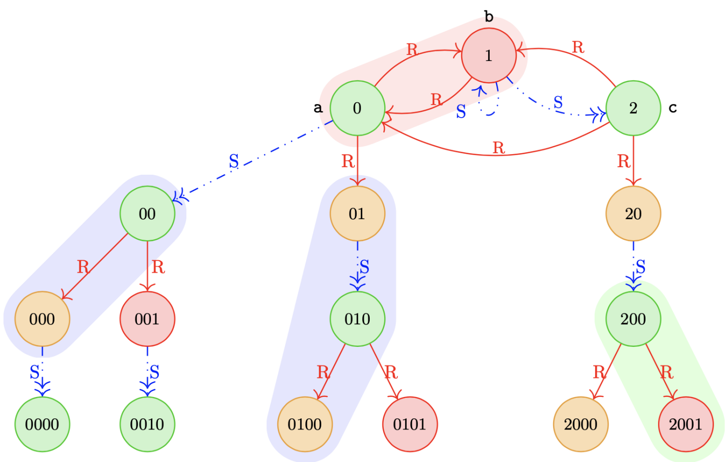
Main ingredients for Querying: Part III (beyond HAF-shaped CQs)

To go beyond HAF-shaped CQs we need an auxiliary notion of a splitting.

Intuitively it mimics a query match by partitioning variables into three sets:

(a) roots, (b) HAFs dangling from roots, and (c) HAFs lying far from roots.

$$q = (\mathrm{A}(x_0) \wedge \mathrm{R}(x_0, x_1) \wedge \mathrm{R}(x_1, x_0) \wedge \mathrm{B}(x_1)) \wedge (\mathrm{S}(x_0, x_{00}) \wedge \mathrm{R}(x_{00}, x_{000})) \wedge (\mathrm{R}(x_0, x_{01}) \wedge \mathrm{S}(x_{01}, x_{010}) \wedge \mathrm{R}(x_{010}, x_{0100})) \wedge (\mathrm{A}(x_{200}) \wedge \mathrm{R}(x_{200}, x_{2001}) \wedge \mathrm{B}(x_{2001}))\,.$$



$$\mathrm{Roots} = \{\, x_0, x_1 \,\}$$

$$\mathrm{SubTree}_1 = \{\, x_{00}, x_{000} \,\}$$

$$\mathrm{SubTree}_2 = \{\, x_{01}, x_{010}, x_{0100} \,\}$$

$$\mathrm{Trees} = \{\, x_{200}, x_{2001} \,\}$$

$$\mathrm{name}(x_0) = \mathtt{a}, \mathrm{name}(x_1) = \mathtt{b}$$

$$\text{root-}\mathit{of}(1) = x_0, \text{root-}\mathit{of}(2) = x_0$$

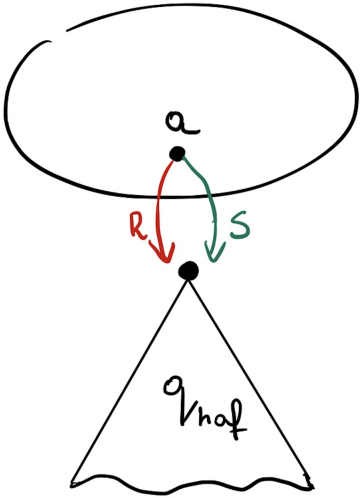
With every splitting Π of q we associate a spoiler an $\mathcal{FGF}$-kb $\mathcal{K}_{\Pi}^{\lightning}$.

Idea: if $\mathcal{K} \cup \mathcal{K}_{\Pi}^{\lightning}$ then there is no matches of q splitting like Π.

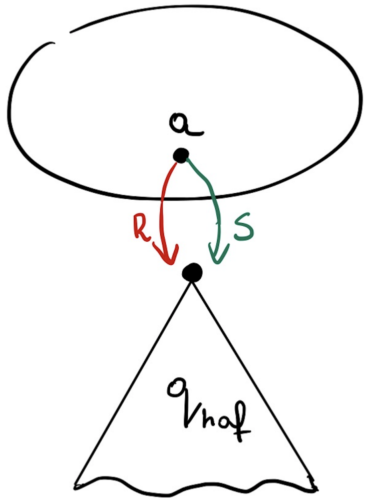
To construct a spoiler we must know how to "describe" Π in $\mathcal{FGF}$, in particular cases (a), (b) and (c).

Main ingredients for Querying: Part IV (detecting rooted HAFs)

Main ingredients for Querying: Part IV (detecting rooted HAFs)



Main ingredients for Querying: Part IV (detecting rooted HAFs)

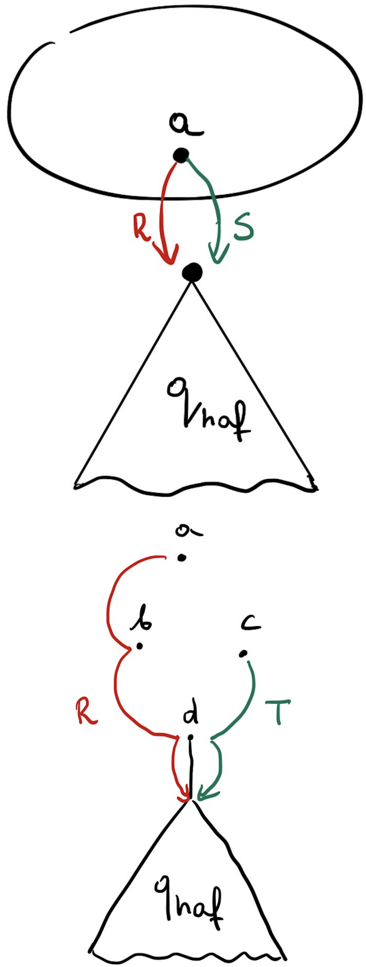


Simply insert

$$(\exists x_2 \, \mathrm{R}(x_1, x_2) \wedge \mathrm{R}(x_1, x_2) \wedge \mathrm{Match}_{q_{\mathsf{haf}}}(x_2)) \, (\mathtt{a})$$

into the DB part of $\mathcal{K}$.

Main ingredients for Querying: Part IV (detecting rooted HAFs)

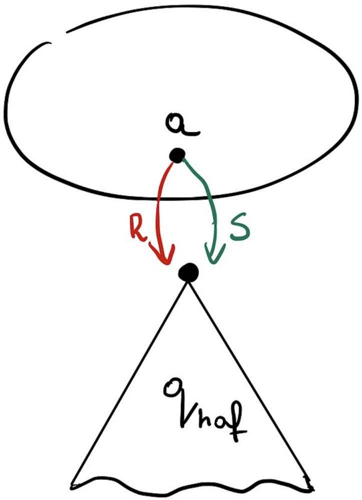


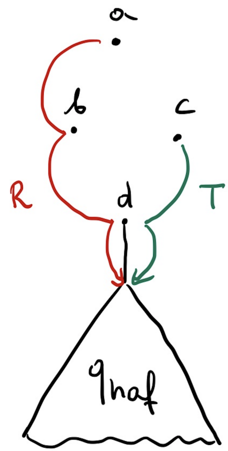
Simply insert

$$(\exists x_2\, \mathrm{R}(x_1, x_2) \wedge \mathrm{R}(x_1, x_2) \wedge \mathrm{Match}_{q_{haf}}(x_2))\,(\mathtt{a})$$

into the DB part of $\mathcal{K}$.

Main ingredients for Querying: Part IV (detecting rooted HAFs)





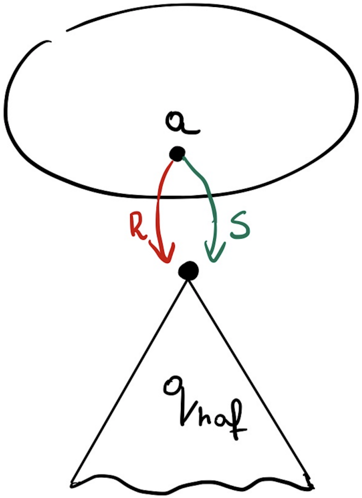
Simply insert

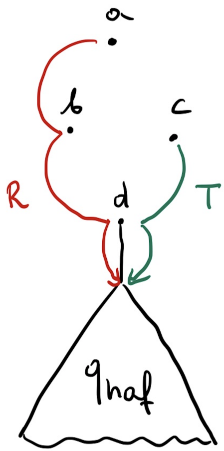
$(\exists x_2\ \mathrm{R}(x_1, x_2) \wedge \mathrm{R}(x_1, x_2) \wedge \mathrm{Match}_{q_{\mathsf{haf}}}(x_2))$ (a)

into the DB part of $\mathcal{K}$.

Fatal error! Not in $\mathcal{FGF}$.

Main ingredients for Querying: Part IV (detecting rooted HAFs)





Simply insert

$$(\exists x_2\ \mathrm{R}(x_1, x_2) \wedge \mathrm{R}(x_1, x_2) \wedge \mathrm{Match}_{q_{\mathrm{haf}}}(x_2))\ (\mathtt{a})$$

into the DB part of $\mathcal{K}$.

Fatal error! Not in $\mathcal{FGF}$.

Repair idea: introduce a bit more constants to $\mathcal{FGF}$ but not too much.

Main ingredients for Querying: Part V (algorithm)

Main ingredients for Querying: Part V (algorithm)

1. We employ a generalisation of spoilers called super-spoilers $\mathcal{K}_q^{\lightning^*}$.

Main ingredients for Querying: Part V (algorithm)

1. We employ a generalisation of spoilers called super-spoilers $\mathcal{K}_q^{\lightning^*}$.

2. If $\mathcal{K} \cup \mathcal{K}_q^{\lightning^*}$ is SAT then $\mathcal{K} \not\models q$.

Main ingredients for Querying: Part V (algorithm)

1. We employ a generalisation of spoilers called super-spoilers $\mathcal{K}_q^{\lightning^*}$.

2. If $\mathcal{K} \cup \mathcal{K}_q^{\lightning^*}$ is SAT then $\mathcal{K} \not\models q$.

3. It turns out that each super-spoiler is of poly-size in $|\mathcal{K}| + |q|$.

Main ingredients for Querying: Part V (algorithm)

1. We employ a generalisation of spoilers called super-spoilers $\mathcal{K}_q^{\lightning^*}$.

2. If $\mathcal{K} \cup \mathcal{K}_q^{\lightning^*}$ is SAT then $\mathcal{K} \not\models q$.

3. It turns out that each super-spoiler is of poly-size in $|\mathcal{K}| + |q|$.

4. There are exponentially many super-spoilers.

Main ingredients for Querying: Part V (algorithm)

1. We employ a generalisation of spoilers called super-spoilers $\mathcal{K}_q^{\lightning *}$.

2. If $\mathcal{K} \cup \mathcal{K}_q^{\lightning *}$ is SAT then $\mathcal{K} \not\models q$.

3. It turns out that each super-spoiler is of poly-size in $|\mathcal{K}| + |q|$.

4. There are exponentially many super-spoilers.

5. Super-spoilers can be enumerated in exponential time.

Main ingredients for Querying: Part V (algorithm)

1. We employ a generalisation of spoilers called super-spoilers $\mathcal{K}_q^{\lightning^*}$.

2. If $\mathcal{K} \cup \mathcal{K}_q^{\lightning^*}$ is SAT then $\mathcal{K} \not\models q$.

3. It turns out that each super-spoiler is of poly-size in $|\mathcal{K}| + |q|$.

4. There are exponentially many super-spoilers.

5. Super-spoilers can be enumerated in exponential time.

6. Hence, we get a reduction to SAT ☺. This also works for unions of CQs.

Main ingredients for Querying: Part V (algorithm)

1. We employ a generalisation of spoilers called super-spoilers $\mathcal{K}_q^{\lightning^*}$.
2. If $\mathcal{K} \cup \mathcal{K}_q^{\lightning^*}$ is SAT then $\mathcal{K} \not\models q$.
3. It turns out that each super-spoiler is of poly-size in $|\mathcal{K}| + |q|$.
4. There are exponentially many super-spoilers.
5. Super-spoilers can be enumerated in exponential time.
6. Hence, we get a reduction to SAT ☺. This also works for unions of CQs.

Theorem

Union of CQs entailment over $\mathcal{FGF}$ knowledge bases is $\textsc{ExpTime}$-complete.

Main ingredients for Querying: Part V (algorithm)

1. We employ a generalisation of spoilers called super-spoilers $\mathcal{K}_q^{\lightning^*}$.

2. If $\mathcal{K} \cup \mathcal{K}_q^{\lightning^*}$ is SAT then $\mathcal{K} \not\models q$.

3. It turns out that each super-spoiler is of poly-size in $|\mathcal{K}| + |q|$.

4. There are exponentially many super-spoilers.

5. Super-spoilers can be enumerated in exponential time.

6. Hence, we get a reduction to SAT ☺. This also works for unions of CQs.

Theorem

Union of CQs entailment over $\mathcal{FGF}$ knowledge bases is $\textsc{ExpTime}$-complete.

Nice application: Forward Guarded Negation fragment of $\mathcal{FO}$

Main ingredients for Querying: Part V (algorithm)

1. We employ a generalisation of spoilers called super-spoilers $\mathcal{K}_q^{\lightning^*}$.

2. If $\mathcal{K} \cup \mathcal{K}_q^{\lightning^*}$ is SAT then $\mathcal{K} \not\models q$.

3. It turns out that each super-spoiler is of poly-size in $|\mathcal{K}| + |q|$.

4. There are exponentially many super-spoilers.

5. Super-spoilers can be enumerated in exponential time.

6. Hence, we get a reduction to SAT ☺. This also works for unions of CQs.

Theorem

Union of CQs entailment over $\mathcal{FGF}$ knowledge bases is $\textsc{ExpTime}$-complete.

Nice application: Forward Guarded Negation fragment of $\mathcal{FO}$

For ψ in (forward) GNFO we poly-compute $\varphi \in$ (forward)GF and a UCQ q s.t.

$$\psi \text{ is SAT iff } \varphi \models q.$$

Main ingredients for Querying: Part V (algorithm)

1. We employ a generalisation of spoilers called super-spoilers $\mathcal{K}_q^{\lightning^*}$.

2. If $\mathcal{K} \cup \mathcal{K}_q^{\lightning^*}$ is SAT then $\mathcal{K} \not\models q$.

3. It turns out that each super-spoiler is of poly-size in $|\mathcal{K}| + |q|$.

4. There are exponentially many super-spoilers.

5. Super-spoilers can be enumerated in exponential time.

6. Hence, we get a reduction to SAT ☺. This also works for unions of CQs.

Theorem

Union of CQs entailment over $\mathcal{FGF}$ knowledge bases is $\textsc{ExpTime}$-complete.

Nice application: Forward Guarded Negation fragment of $\mathcal{FO}$

For ψ in (forward) GNFO we poly-compute $\varphi \in$ (forward)GF and a UCQ q s.t.

$$\psi \text{ is SAT iff } \varphi \models q.$$

Theorem

The satisfiability of Forward Guarded Negation $\mathcal{FO}$ is $\textsc{ExpTime}$-complete.

Conclusions

Conclusions

Forward GF = formulae guarded but kept forward

Conclusions

Forward GF = formulae guarded but kept forward

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete, also in the finite. Data complexity (co)NP-complete. FMP + Fin-Control.

Conclusions

Forward GF = formulae guarded but kept forward

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete, also in the finite. Data complexity (co)NP-complete. FMP + Fin-Control.

Open problems and future research?

Conclusions

Forward GF = formulae guarded but kept forward

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete, also in the finite. Data complexity (co)NP-complete. FMP + Fin-Control.

Open problems and future research?

1. Understand model theory of Ordered/Fluted/Forward Fragment of $\mathcal{FO}$.

Conclusions

Forward GF = formulae guarded but kept forward

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete, also in the finite. Data complexity (co)NP-complete. FMP + Fin-Control.

Open problems and future research?

1. Understand model theory of Ordered/Fluted/Forward Fragment of $\mathcal{FO}$.

i.e. E-F Games, Craig Interpolation, Beth Definability, Preservation Theorems à la Łoś-Tarski

Conclusions

Forward GF = formulae guarded but kept forward

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete, also in the finite. Data complexity (co)NP-complete. FMP + Fin-Control.

Open problems and future research?

1. Understand model theory of Ordered/Fluted/Forward Fragment of $\mathcal{FO}$.

i.e. E-F Games, Craig Interpolation, Beth Definability, Preservation Theorems à la Łoś-Tarski

Ongoing work with Reijo Jaakkola, University of Tampere

Conclusions

Forward GF = formulae guarded but kept forward

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete, also in the finite. Data complexity (co)NP-complete. FMP + Fin-Control.

Open problems and future research?

1. Understand model theory of Ordered/Fluted/Forward Fragment of $\mathcal{FO}$.

i.e. E-F Games, Craig Interpolation, Beth Definability, Preservation Theorems à la Łoś-Tarski

Ongoing work with Reijo Jaakkola, University of Tampere

2. Study $\mathcal{FGF} + \mathcal{I}/\mathcal{O}/\mathcal{Q}$ (partial results obtained)

Conclusions

Forward GF = formulae guarded but kept forward

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete, also in the finite. Data complexity (co)NP-complete. FMP + Fin-Control.

Open problems and future research?

1. Understand model theory of Ordered/Fluted/Forward Fragment of $\mathcal{FO}$.

i.e. E-F Games, Craig Interpolation, Beth Definability, Preservation Theorems à la Łoś-Tarski

Ongoing work with Reijo Jaakkola, University of Tampere

2. Study $\mathcal{FGF} + \mathcal{I}/\mathcal{O}/\mathcal{Q}$ (partial results obtained)

3. $\mathcal{FGF}+\mu$ or $\mathcal{FGF}+\mathcal{S}$ behave nicer than $\mathcal{GF}+TG$ (with E. Kieronski)

Conclusions

Forward GF = formulae guarded but kept forward

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete, also in the finite. Data complexity (co)NP-complete. FMP + Fin-Control.

Open problems and future research?

1. Understand model theory of Ordered/Fluted/Forward Fragment of $\mathcal{FO}$.

i.e. E-F Games, Craig Interpolation, Beth Definability, Preservation Theorems à la Łoś-Tarski

Ongoing work with Reijo Jaakkola, University of Tampere

2. Study $\mathcal{FGF} + \mathcal{I}/\mathcal{O}/\mathcal{Q}$ (partial results obtained)

3. $\mathcal{FGF}+\mu$ or $\mathcal{FGF}+\mathcal{S}$ behave nicer than $\mathcal{GF}+\mathcal{TG}$ (with E. Kieronski)

4. Effective algorithms, e.g. resolution-based/sequent proofs (with Tim Lyon).

Conclusions

Forward GF = formulae guarded but kept forward

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete, also in the finite. Data complexity (co)NP-complete. FMP + Fin-Control.

Open problems and future research?

1. Understand model theory of Ordered/Fluted/Forward Fragment of $\mathcal{FO}$.

i.e. E-F Games, Craig Interpolation, Beth Definability, Preservation Theorems à la Łoś-Tarski

Ongoing work with Reijo Jaakkola, University of Tampere

2. Study $\mathcal{FGF} + \mathcal{I}/\mathcal{O}/\mathcal{Q}$ (partial results obtained)

3. $\mathcal{FGF}+\mu$ or $\mathcal{FGF}+\mathcal{S}$ behave nicer than $\mathcal{GF}+TG$ (with E. Kieronski)

4. Effective algorithms, e.g. resolution-based/sequent proofs (with Tim Lyon).

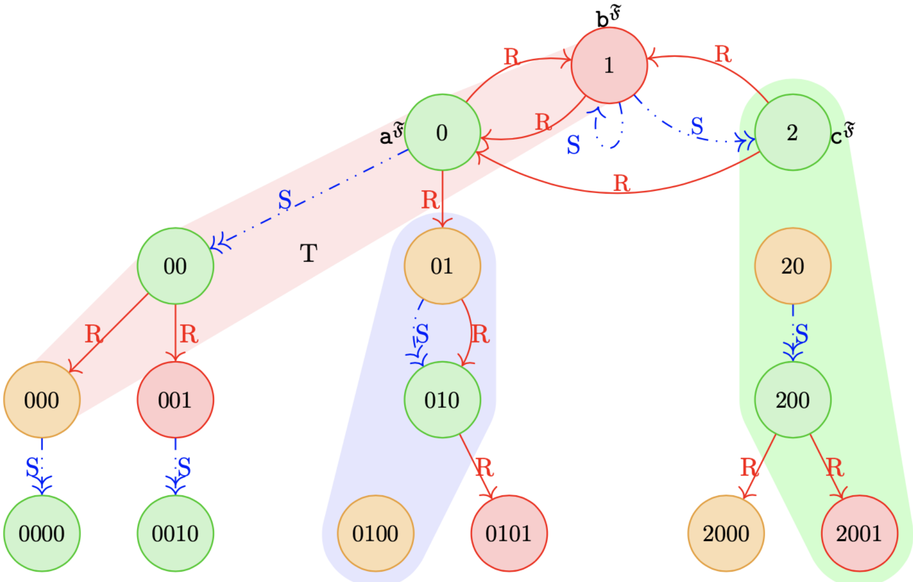
5. Forward TGDs (with Piotr Nalewaja).

Conclusions

Forward GF = formulae guarded but kept forward

Theorem (B., JELIA 2021)

Knowledge-base SAT and CQ entailment for $\mathcal{FGF}$ is $\textsc{ExpTime}$-complete, also in the finite. Data complexity (co)NP-complete. FMP + Fin-Control.



research?

ward Fragment of $\mathcal{FO}$.

ervation Theorems à la Łoś-Tarski

rsity of Tampere

)

G (with E. Kieronski)

nt proofs (with Tim Lyon).

Thanks for attention!