Modulo Counting on Words and Trees

(joint work with Witold Charatonik)

école normale supérieure paris—saclay

Uniwersytet Wrocławski

Bartosz Bednarczyk

bartosz.bednarczyk@ens-paris-saclay.fr

École normale supérieure Paris-Saclay
and University of Wrocław

FSTTCS 2017
Kanpur, December 13, 2017

Agenda

- Classical results on $\mathcal{FO}^2$ and related logics
- Logics on restricted classes of structures (words and trees)
- The main results of the paper
  - namely the exact complexity of nice family of tree logics
  - able to handle modulo constraints (like parity)
  - with relatively small complexity blowup
- Proof ideas
- Our current research and open problems

Historical results

Facts about SAT and $\mathcal{FO}^2$ on arbitrary structures

- We are interested in finite satisfiability problems
- Models = purely relational structures, no constants, no functions

Facts about SAT and $\mathcal{FO}^2$ on arbitrary structures

- We are interested in finite satisfiability problems
- Models = purely relational structures, no constants, no functions
- Some classical results:
  - $\mathcal{FO}$ undecidable (Church, Turing; 1930s)

Facts about SAT and $\mathcal{FO}^2$ on arbitrary structures

- We are interested in finite satisfiability problems
- Models = purely relational structures, no constants, no functions
- Some classical results:
  - $\mathcal{FO}$ undecidable (Church, Turing; 1930s)
  - $\mathcal{FO}^3$ undecidable (Kahr, Moore, Wang; 1959)

Facts about SAT and $\mathcal{FO}^2$ on arbitrary structures

- We are interested in finite satisfiability problems
- Models = purely relational structures, no constants, no functions
- Some classical results:
  - $\mathcal{FO}$ undecidable (Church, Turing; 1930s)
  - $\mathcal{FO}^3$ undecidable (Kahr, Moore, Wang; 1959)
  - $\mathcal{FO}^2$ decidable (Mortimer; 1975)
  - $\mathcal{FO}^2$ enjoys exponential model property (Gradel, Kolaitis, Vardi; 1997) - NExpTime-completeness

Facts about SAT and $\mathcal{FO}^2$ on arbitrary structures

- We are interested in finite satisfiability problems
- Models = purely relational structures, no constants, no functions
- Some classical results:
  - $\mathcal{FO}$ undecidable (Church, Turing; 1930s)
  - $\mathcal{FO}^3$ undecidable (Kahr, Moore, Wang; 1959)
  - $\mathcal{FO}^2$ decidable (Mortimer; 1975)
  - $\mathcal{FO}^2$ enjoys exponential model property (Gradel, Kolaitis, Vardi; 1997) - NEXPTIME-completeness
  - Connection between $\mathcal{FO}^2$ and modal, temporal, description logics;
  - many applications in verification and databases

Facts about SAT and $\mathcal{FO}^2$ on arbitrary structures

- We are interested in finite satisfiability problems
- Models = purely relational structures, no constants, no functions
- Some classical results:
  - $\mathcal{FO}$ undecidable (Church, Turing; 1930s)
  - $\mathcal{FO}^3$ undecidable (Kahr, Moore, Wang; 1959)
  - $\mathcal{FO}^2$ decidable (Mortimer; 1975)
  - $\mathcal{FO}^2$ enjoys exponential model property (Gradel, Kolaitis, Vardi; 1997) - NEXPTIME-completeness
  - Connection between $\mathcal{FO}^2$ and modal, temporal, description logics;
  - many applications in verification and databases

Example formula:
from each element there exists a path of length 3

$$\forall x \exists y\, (E(x, y) \wedge \exists x\, (E(y, x) \wedge \exists y\; E(x, y)))$$

Facts about SAT and $\mathcal{FO}^2$ on arbitrary structures

- We are interested in finite satisfiability problems
- Models = purely relational structures, no constants, no functions
- Some classical results:
  - $\mathcal{FO}$ undecidable (Church, Turing; 1930s)
  - $\mathcal{FO}^3$ undecidable (Kahr, Moore, Wang; 1959)
  - $\mathcal{FO}^2$ decidable (Mortimer; 1975)
  - $\mathcal{FO}^2$ enjoys exponential model property (Gradel, Kolaitis, Vardi; 1997) - NExpTime-completeness
  - Connection between $\mathcal{FO}^2$ and modal, temporal, description logics;
  - many applications in verification and databases

Example formula:
from each element there exists a path of length 3

$$\forall x \exists y\, (E(x, y) \wedge \exists x\, (E(y, x) \wedge \exists y\; E(x, y)))$$

Conclusion: $\mathcal{FO}^2$ decidable, but limited in terms of expressivity.

Logics on trees

Possible variations

There are several scenarios which may influence decidability/complexity. E.g., we may consider:

Possible variations

There are several scenarios which may influence decidability/complexity. E.g., we may consider:

- Ordered vs Unordered trees

Possible variations

There are several scenarios which may influence decidability/complexity. E.g., we may consider:

- Ordered vs Unordered trees
- Ranked vs Unranked trees

Possible variations

There are several scenarios which may influence decidability/complexity. E.g., we may consider:

- Ordered vs Unordered trees
- Ranked vs Unranked trees
- Finite vs Infinite trees

Possible variations

There are several scenarios which may influence decidability/complexity. E.g., we may consider:

- Ordered vs Unordered trees
- Ranked vs Unranked trees
- Finite vs Infinite trees
- With unary alphabet restriction (UAR) or without UAR
  - precisely one unary predicate holds at each node
- ...

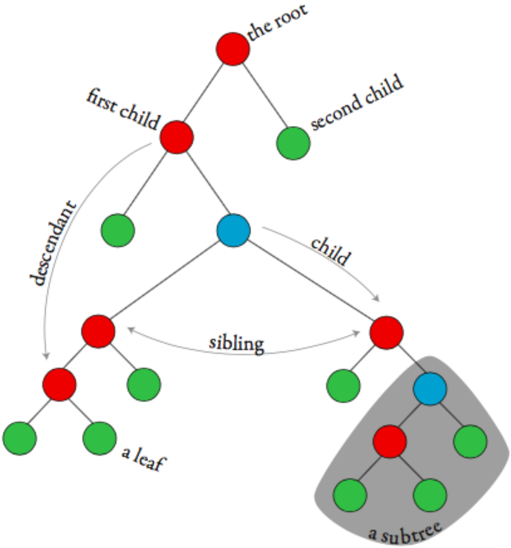
Possible variations

There are several scenarios which may influence decidability/complexity. E.g., we may consider:

- Ordered vs Unordered trees
- Ranked vs Unranked trees
- Finite vs Infinite trees
- With unary alphabet restriction (UAR) or without UAR
  - precisely one unary predicate holds at each node
- . . .

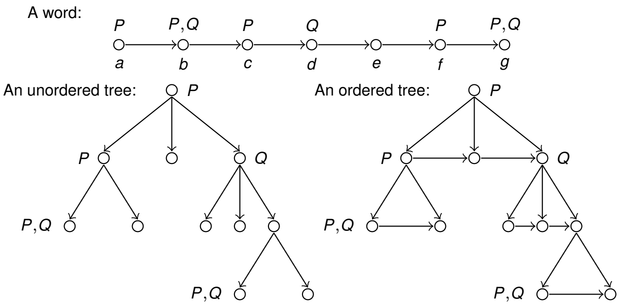
We will focus on Finite, Ordered, Unranked Trees, where multiple predicates can hold at one node (without UAR).

Tree notions



Signature $\tau = \tau_0 \cup \tau_{nav}$

- τ_0 – unary symbols (usually P, Q, etc.)
- τ_{nav} – navigational binary symbols with fixed interpretation
  - words: $\leq$ (order over positions), $+1$ (it's induced successor)
  - unordered trees: $\downarrow$ (child), $\downarrow_+$ (descendant, TC of $\downarrow$)
  - ordered trees: $\downarrow$, $\downarrow_+$, $\rightarrow$ (next sibling), $\rightarrow^+$ (TC of $\rightarrow$)



Complexity results

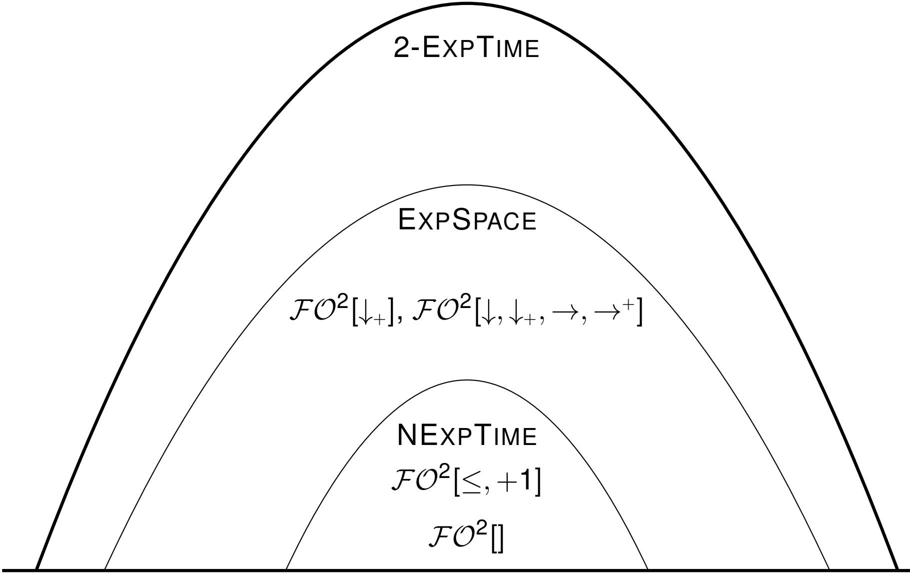
- $\mathcal{FO}$ is TOWER-complete, even for $\mathcal{FO}^3$ (Stockmeyer; 1974).

Complexity results

- $\mathcal{FO}$ is TOWER-complete, even for $\mathcal{FO}^3$ (Stockmeyer; 1974).
- $\mathcal{FO}^2[\leq, +1]$ on finite words
  - $\mathcal{FO}^2$ is NEXPTIME-complete (Etessami et al, LICS 1997)
  - Equally expressive to Unary Temporal Logic
  - $\mathcal{FO}^2 + \exists^{\leq k} + \exists^{\geq k}$ still in NEXPTIME (Charatonik et al, CSL 2015)

Complexity results

- $\mathcal{FO}$ is TOWER-complete, even for $\mathcal{FO}^3$ (Stockmeyer; 1974).
- $\mathcal{FO}^2[\leq, +1]$ on finite words
  - $\mathcal{FO}^2$ is NEXPTIME-complete (Etessami et al, LICS 1997)
  - Equally expressive to Unary Temporal Logic
  - $\mathcal{FO}^2 + \exists^{\leq k} + \exists^{\geq k}$ still in NEXPTIME (Charatonik et al, CSL 2015)
- $\mathcal{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ on finite trees
  - $\mathcal{FO}^2$ on trees is EXPSPACE-complete (Benaim et al, ICALP 2013).
  - Equally expressive to Navigational XPath (Marx et al, 2004).
  - $\mathcal{FO}^2 + \exists^{\leq k} + \exists^{\geq k}$ still in EXPSPACE (Bednarczyk et al, CSL 2017)


2-ExpTime
ExpSpace
$\mathcal{FO}^2[\downarrow_+]$, $\mathcal{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$
NExpTime
$\mathcal{FO}^2[\leq, +1]$
$\mathcal{FO}^2[]$

Our results

Our settings

We work on extensions of $\mathcal{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ and $\mathcal{FO}^2[\leq, +1]$.

Logics with modulo

$$\mathcal{FO}^2_{\mathrm{MOD}} = \mathcal{FO}^2 + \exists^{=k (\text{mod } l)}$$

for arbitrary natural numbers k, l written in binary

Our contribution

$\mathcal{FO}^2_{\mathrm{MOD}}$ on words

An alternative proof of $\mathcal{FO}^2_{\mathrm{MOD}}[\leq, +1]$ EXPSPACE-upper bound.

Our contribution

$\mathcal{FO}^2_{\mathrm{MOD}}$ on words

An alternative proof of $\mathcal{FO}^2_{\mathrm{MOD}}[\leq, +1]$ EXPSPACE-upper bound.

Theorem ($\mathcal{FO}^2_{\mathrm{MOD}}$ on trees - upper bound)

Membership of $\mathcal{FO}^2_{\mathrm{MOD}}[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ to 2-EXPTIME.

Theorem ($\mathcal{FO}^2_{\mathrm{MOD}}$ on trees - lower bound)

2-EXPTIME-hardness for $\mathcal{FO}^2_{\mathrm{MOD}}[\downarrow, \downarrow_+]$.

Why modulo counting matters?

The general idea of counting quantifiers in logic

- Goal: increase expressiveness by adding an ability to count
  - Counting quantifiers $\exists^{\geq k}, \exists^{\leq k}$
  - Graded modalities $\Diamond_{\geq k}, \Diamond_{\leq k}, E^{\geq k}, A^{\leq k}$

The general idea of counting quantifiers in logic

- Goal: increase expressiveness by adding an ability to count
  - Counting quantifiers $\exists^{\geq k}, \exists^{\leq k}$
  - Graded modalities $\Diamond_{\geq k}, \Diamond_{\leq k}$, $E^{\geq k}, A^{\leq k}$
- Well-known extensions:
  - Graded modal logic (over 25 papers!, 1985 - ...)
  - Graded PDL (Nguyen, CS&P 2015)
  - Graded strategy logic, CTL, CTL* (Murano et al, 2010-2016)
  - Graded μ-calculus (Kupferman et al, CADE 2002)
  - $\mathcal{FO}^2$ and $\mathcal{GF}^2$ with counting quantifiers (Pratt-Hartmann 2007)
  - $\mathcal{FO}^2$ with counting on words (Charatonik et al, CSL 2015)
  - $\mathcal{FO}^2$ with counting on trees (Bednarczyk et al, CSL 2017)
  - description logics, dependence logic, epistemic logic
  - and so on, and so on, and so on...

The general idea of counting quantifiers in logic

- Goal: increase expressiveness by adding an ability to count
  - Counting quantifiers $\exists^{\geq k}, \exists^{\leq k}$
  - Graded modalities $\Diamond_{\geq k}, \Diamond_{\leq k}$, $E^{\geq k}, A^{\leq k}$
- Well-known extensions:
  - Graded modal logic (over 25 papers!, 1985 - ...)
  - Graded PDL (Nguyen, CS&P 2015)
  - Graded strategy logic, CTL, CTL* (Murano et al, 2010-2016)
  - Graded μ-calculus (Kupferman et al, CADE 2002)
  - $\mathcal{FO}^2$ and $\mathcal{GF}^2$ with counting quantifiers (Pratt-Hartmann 2007)
  - $\mathcal{FO}^2$ with counting on words (Charatonik et al, CSL 2015)
  - $\mathcal{FO}^2$ with counting on trees (Bednarczyk et al, CSL 2017)
  - description logics, dependence logic, epistemic logic
  - and so on, and so on, and so on...
- This talk: What if we change a little the way we count?

Why modulo counting matters?

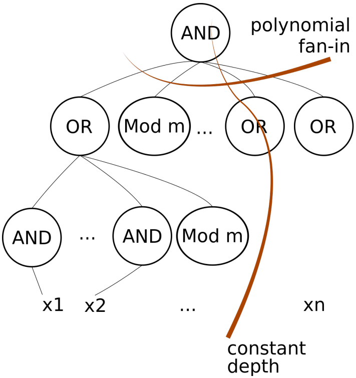
- Parity - not expressible in $\mathcal{FO}$

Why modulo counting matters?

- Parity - not expressible in $\mathcal{FO}$
- Definability is well-studied on words and trees (Straubing 2008 survey), but satisfiability was neglected

Why modulo counting matters?

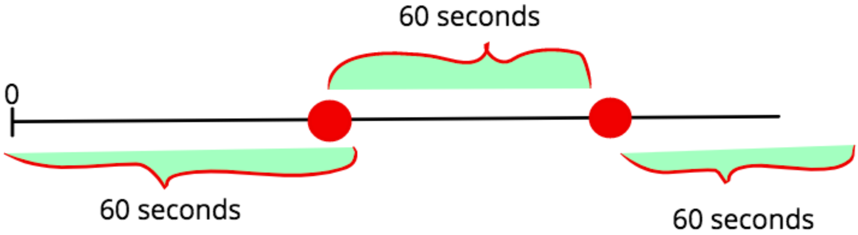
- Parity - not expressible in $\mathcal{FO}$
- Definability is well-studied on words and trees (Straubing 2008 survey), but satisfiability was neglected
- Connections with circuit complexity
  - PARITY is not in AC^0
  - Modular gates + AC^0 = ACC^0
  - Separating NC^1 from ACC^0 (important open problem!)



Why modulo counting matters?

An example property expressible in $\mathcal{FO}^2_{\mathrm{MOD}}$

There is an alarm every 60 seconds.



$$\forall x \left(\left(\exists^{=0 (\mathrm{mod}\ 60)} y (y < x) \right) \rightarrow alarm(x) \right)$$

Proof ideas - lower bound

Lower bound

How did we prove the lower bound?

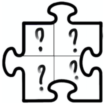
- We introduced a new version of tilling games
- 2-EXPTIME-compl by painful reduction from halting for AEXPSPACE Turning machines
- Encoding of winning strategy of game in our logic

Tilling game

Prover and Spoiler

Rules

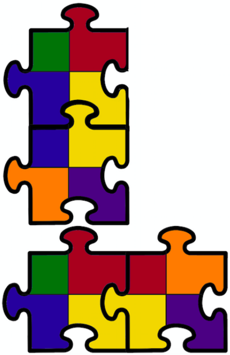
- Finite set of puzzles



- Horizontal and vertical constraints
- Goal: Construct a correct tilling of a board of the size $2^n \times k$

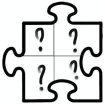
Tilling game

Constraints



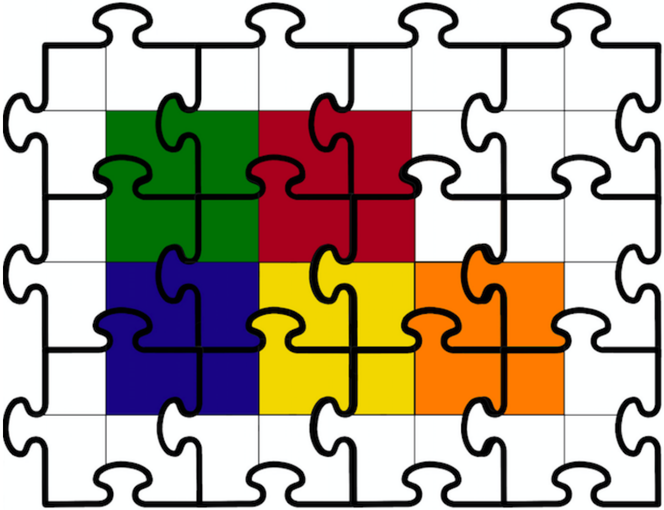
Rules

- Finite set of puzzles

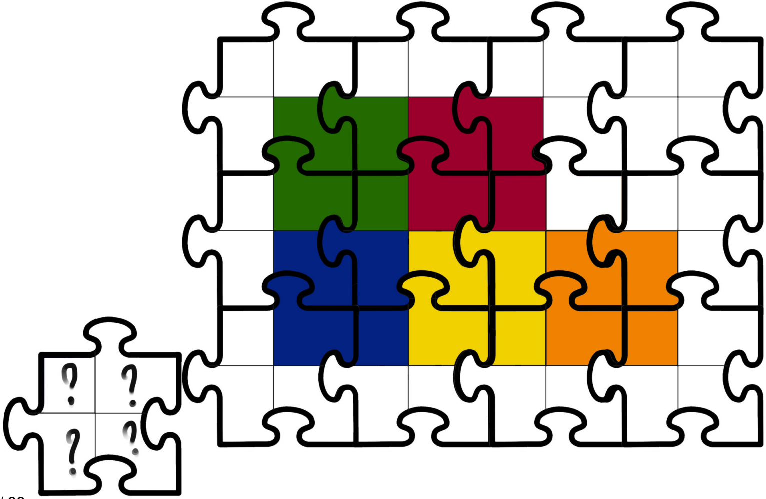


- Horizontal and vertical constraints
- Goal: Construct a correct tilling of a board of the size $2^n \times k$

An example: Correct tilling



How modulo counting help us to play this game?



Proof ideas - upper bound

Finite satisfiability cooking recipe

Finite satisfiability cooking recipe

- Step 1. Transform your formula into a normal form

Finite satisfiability cooking recipe

- Step 1. Transform your formula into a normal form
- Step 2. Design a right notion of a type
  - And prove that your notion is ”correct” . . .

Finite satisfiability cooking recipe

- Step 1. Transform your formula into a normal form
- Step 2. Design a right notion of a type
  - And prove that your notion is "correct" . . .
- Step 3. Show small model property
  - Restrict your attention to trees with:
    - doubly-exponential degree of every nodes and
    - doubly-exponentially long paths
  - Do it by cutting out too long $\downarrow$ and $\rightarrow$-paths

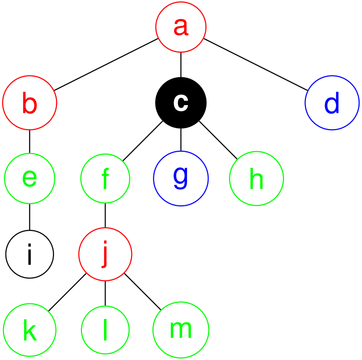
Finite satisfiability cooking recipe

- Step 1. Transform your formula into a normal form
- Step 2. Design a right notion of a type
  - And prove that your notion is ”correct” . . .
- Step 3. Show small model property
  - Restrict your attention to trees with:
    - doubly-exponential degree of every nodes and
    - doubly-exponentially long paths
  - Do it by cutting out too long $\downarrow$ and $\rightarrow$-paths
- Step 4. Present an alternating algorithm
  - in this case AEXPSPACE (= 2-EXPTIME)

Order formulas

Assuming $\tau_{nav} = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$.
There are ten of them:

Position Θ	Θ–related with "c"
$\theta_=$	
$\theta_\downarrow$	
$\theta_\uparrow$	
$\theta_{\downarrow\downarrow^+}$	
$\theta_\rightarrow$	
$\theta_\leftarrow$	
$\theta_{\uparrow\uparrow^+}$, $\theta_{\rightrightarrows^+}$, $\theta_{\leftleftarrows^+}$	
$\theta_{\not\sim}$	

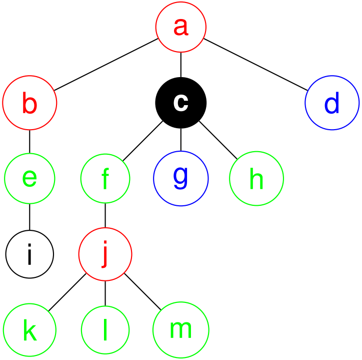


Ex: $\theta_{\downarrow\downarrow_+}(x, y) = x{\downarrow_+}y \wedge \neg(x{\downarrow}y)$

Order formulas

Assuming $\tau_{nav} = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$.
There are ten of them:

Position Θ	Θ–related with "c"
$\theta_=$	$\{c\}$
$\theta_\downarrow$	
$\theta_\uparrow$	
$\theta_{\downarrow\downarrow^+}$	
$\theta_\rightarrow$	
$\theta_\leftarrow$	
$\theta_{\uparrow\uparrow^+}$, $\theta_{\rightrightarrows^+}$, $\theta_{\leftleftarrows^+}$	
$\theta_{\not\sim}$	

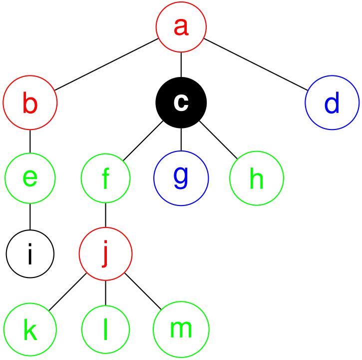


Ex: $\theta_{\downarrow\downarrow_+}(x, y) = x{\downarrow_+}y \wedge \neg(x{\downarrow}y)$

Order formulas

Assuming $\tau_{nav} = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$.
There are ten of them:

Position Θ	Θ–related with "c"
$\theta_=$	$\{c\}$
$\theta_\downarrow$	$\{f, g, h\}$
$\theta_\uparrow$	
$\theta_{\downarrow\downarrow_+}$	
$\theta_\rightarrow$	
$\theta_\leftarrow$	
$\theta_{\uparrow\uparrow^+}$, $\theta_{\rightrightarrows^+}$, $\theta_{\leftleftarrows^+}$	
$\theta_{\not\sim}$	

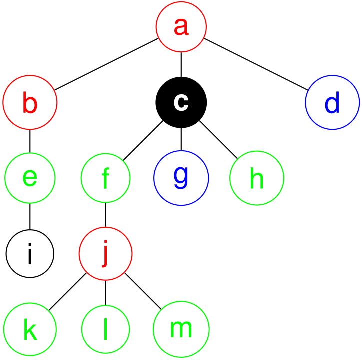


Ex: $\theta_{\downarrow\downarrow_+}(x, y) = x{\downarrow_+}y \wedge \neg(x{\downarrow}y)$

Order formulas

Assuming $\tau_{nav} = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$.
There are ten of them:

Position Θ	Θ–related with "c"
$\theta_=$	$\{c\}$
$\theta_\downarrow$	$\{f, g, h\}$
$\theta_\uparrow$	$\{a\}$
$\theta_{\downarrow\downarrow_+}$	
$\theta_\rightarrow$	
$\theta_\leftarrow$	
$\theta_{\uparrow\uparrow^+}$, $\theta_{\rightrightarrows^+}$, $\theta_{\leftleftarrows^+}$	
$\theta_{\not\sim}$	

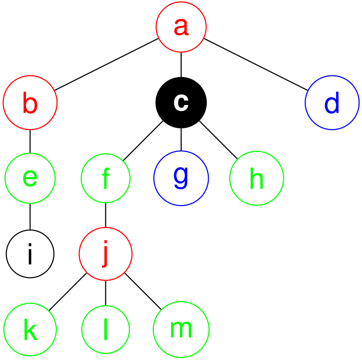


Ex: $\theta_{\downarrow\downarrow_+}(x, y) = x{\downarrow_+}y \wedge \neg(x{\downarrow}y)$

Order formulas

Assuming $\tau_{nav} = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$.
There are ten of them:

Position Θ	Θ–related with "c"
$\theta_=$	$\{c\}$
$\theta_\downarrow$	$\{f, g, h\}$
$\theta_\uparrow$	$\{a\}$
$\theta_{\downarrow\downarrow_+}$	$\{j, k, l, m\}$
$\theta_\rightarrow$	
$\theta_\leftarrow$	
$\theta_{\uparrow\uparrow^+}$, $\theta_{\rightrightarrows^+}$, $\theta_{\leftleftarrows^+}$	
$\theta_{\not\sim}$	

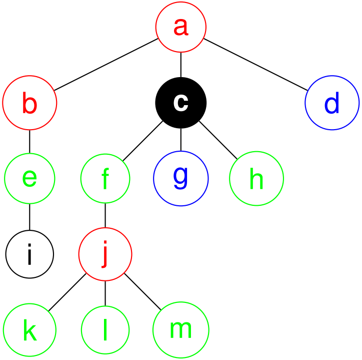


Ex: $\theta_{\downarrow\downarrow_+}(x, y) = x{\downarrow_+}y \wedge \neg(x{\downarrow}y)$

Order formulas

Assuming $\tau_{nav} = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$.
There are ten of them:

Position Θ	Θ–related with "c"
$\theta_=$	$\{c\}$
$\theta_\downarrow$	$\{f, g, h\}$
$\theta_\uparrow$	$\{a\}$
$\theta_{\downarrow\downarrow^+}$	$\{j, k, l, m\}$
$\theta_\rightarrow$	$\{d\}$
$\theta_\leftarrow$	
$\theta_{\uparrow\uparrow^+}$, $\theta_{\rightrightarrows^+}$, $\theta_{\leftleftarrows^+}$	
$\theta_{\not\sim}$	

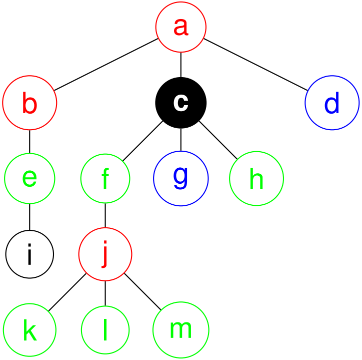


Ex: $\theta_{\downarrow\downarrow_+}(x, y) = x\downarrow_+ y \wedge \neg(x\downarrow y)$

Order formulas

Assuming $\tau_{nav} = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$.
There are ten of them:

Position Θ	Θ–related with "c"
$\theta_=$	$\{c\}$
$\theta_\downarrow$	$\{f, g, h\}$
$\theta_\uparrow$	$\{a\}$
$\theta_{\downarrow\downarrow_+}$	$\{j, k, l, m\}$
$\theta_\rightarrow$	$\{d\}$
$\theta_\leftarrow$	$\{b\}$
$\theta_{\uparrow\uparrow^+}, \theta_{\rightrightarrows^+}, \theta_{\leftleftarrows^+}$	
$\theta_{\not\sim}$	

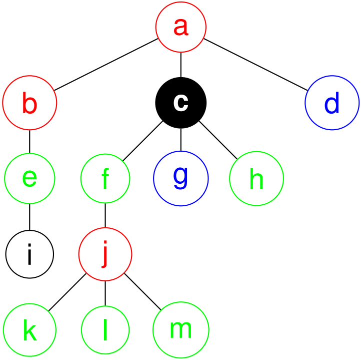


Ex: $\theta_{\downarrow\downarrow_+}(x, y) = x\downarrow_+ y \wedge \neg(x\downarrow y)$

Order formulas

Assuming $\tau_{nav} = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$.
There are ten of them:

Position Θ	Θ–related with "c"
$\theta_=$	$\{c\}$
$\theta_\downarrow$	$\{f, g, h\}$
$\theta_\uparrow$	$\{a\}$
$\theta_{\downarrow\downarrow_+}$	$\{j, k, l, m\}$
$\theta_\rightarrow$	$\{d\}$
$\theta_\leftarrow$	$\{b\}$
$\theta_{\uparrow\uparrow^+}$, $\theta_{\rightrightarrows^+}$, $\theta_{\leftleftarrows^+}$	$\emptyset$
$\theta_{\not\sim}$	

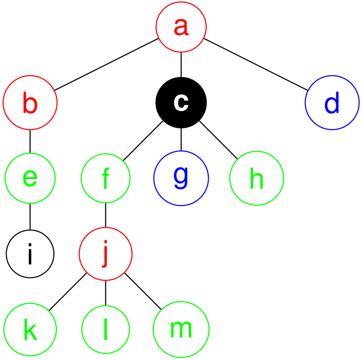


Ex: $\theta_{\downarrow\downarrow_+}(x, y) = x\downarrow_+ y \wedge \neg(x\downarrow y)$

Order formulas

Assuming $\tau_{nav} = \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$.
There are ten of them:

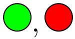
Position Θ	Θ–related with "c"
$\theta_=$	$\{c\}$
$\theta_\downarrow$	$\{f, g, h\}$
$\theta_\uparrow$	$\{a\}$
$\theta_{\downarrow\downarrow_+}$	$\{j, k, l, m\}$
$\theta_\rightarrow$	$\{d\}$
$\theta_\leftarrow$	$\{b\}$
$\theta_{\uparrow\uparrow^+}$, $\theta_{\rightrightarrows^+}$, $\theta_{\leftleftarrows^+}$	$\emptyset$
$\theta_{\not\sim}$	$\{e, i\}$

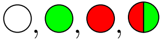


Ex: $\theta_{\downarrow\downarrow_+}(x, y) = x\downarrow_+ y \wedge \neg(x\downarrow y)$

Atomic 1-types

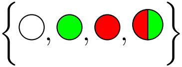
- 1-type over signature τ is a color of a single node
- The total number of 1-types is bounded exponentially in $|\tau|$
- Example:

Unary symbols $\tau_0 = \Big\{$  $\Big\} = \Big\{\text{Green}(), \text{Red}()\Big\}$

Possible 1-types $\boldsymbol{\alpha}_{\tau_0} = \Big\{$  $\Big\}$

A new ingredient - Full type - definition

- Recall that:
  - 1-types $\boldsymbol{\alpha}_{\tau_0}$ - colors of nodes over signature τ_0

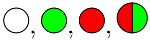
An example: $\boldsymbol{\alpha}_{\tau_0} =$ 

  - Positions Θ - how to compare nodes

$$\Theta = \{\theta_{=}, \theta_{\downarrow}, \theta_{\uparrow}, \theta_{\downarrow\downarrow^{+}}, \theta_{\uparrow\uparrow^{+}}, \theta_{\rightarrow}, \theta_{\leftarrow}, \theta_{\rightrightarrows^{+}}, \theta_{\leftleftarrows^{+}}, \theta_{\not\sim}\}$$

A new ingredient - Full type - definition

- Recall that:
  - 1-types $\boldsymbol{\alpha}_{\tau_0}$ - colors of nodes over signature τ_0

An example: $\boldsymbol{\alpha}_{\tau_0} = \Big\{$  $\Big\}$

  - Positions Θ - how to compare nodes

$$\Theta = \{\theta_{=}, \theta_{\downarrow}, \theta_{\uparrow}, \theta_{\downarrow\downarrow^{+}}, \theta_{\uparrow\uparrow^{+}}, \theta_{\rightarrow}, \theta_{\leftarrow}, \theta_{\rightrightarrows^{+}}, \theta_{\leftleftarrows^{+}}, \theta_{\not\sim}\}$$

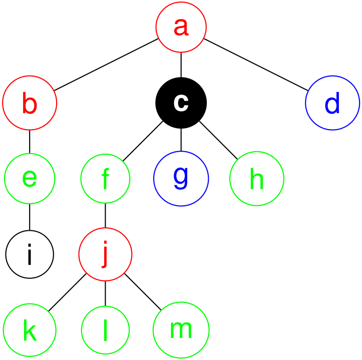
- $(\mathbb{Z}_{l_1}, \ldots, \mathbb{Z}_{l_n})$–Full type
  - information about the whole tree from local point of view

$$(\mathbb{Z}_{l_1}, \ldots, \mathbb{Z}_{l_n})\text{-}\mathit{ftp}(x) :: \Theta \to \boldsymbol{\alpha} \to \{0, 1\} \times \mathbb{Z}_{l_1} \times \ldots \times \mathbb{Z}_{l_n}$$

  - The total number of ftps is doubly-exponential.

A full type example

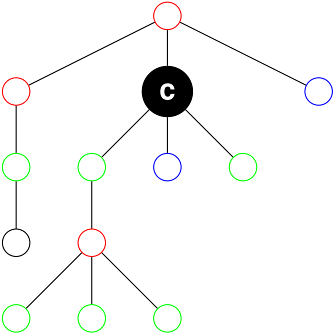
$$\boldsymbol{\alpha} = \left\{ \bullet, \bullet, \bullet, \bullet \right\} \quad \mathit{ftp}(c) : \Theta \to \boldsymbol{\alpha} \to \{0,1\} \times \mathbb{Z}_3$$



Θ	●	●	●	●
$\theta_=$	?	?	?	?
$\theta_\downarrow$	?	?	?	?
$\theta_{\downarrow\downarrow+}$	?	?	?	?
$\theta_{\not\nearrow}$	...	...	...	...
$\theta_\to$	...	...	...	...
...				

A full type example

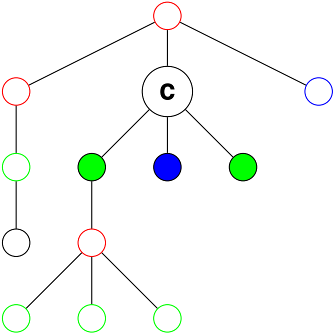
$$\boldsymbol{\alpha} = \left\{ \textcolor{blue}{\bullet}, \textcolor{red}{\bullet}, \textcolor{green}{\bullet}, \bullet \right\} \quad ftp(c) : \Theta \to \boldsymbol{\alpha} \to \{0, 1\} \times \mathbb{Z}_3$$



Θ	$\textcolor{blue}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{green}{\bullet}$	$\bullet$
$\theta_{=}$	?	?	?	?
$\theta_{\downarrow}$	?	?	?	?
$\theta_{\downarrow\downarrow^{+}}$	?	?	?	?
$\theta_{\not\sim}$	...	...	...	...
$\theta_{\rightarrow}$	...	...	...	...
...				

A full type example

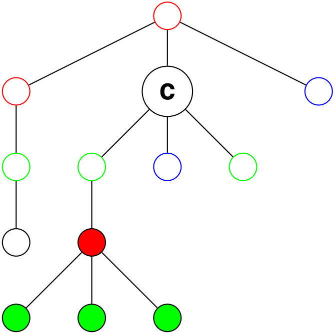
$$\boldsymbol{\alpha} = \{\bullet, \bullet, \bullet, \bullet\} \quad ftp(c) : \Theta \to \boldsymbol{\alpha} \to \{0, 1\} \times \mathbb{Z}_3$$



Θ	●	●	●	●
$\theta_{=}$	(0,0)	(0,0)	(0,0)	(1,1)
$\theta_{\downarrow}$	?	?	?	?
$\theta_{\downarrow\downarrow^{+}}$	?	?	?	?
$\theta_{\not\sim}$	...	...	...	...
$\theta_{\rightarrow}$	...	...	...	...
...				

A full type example

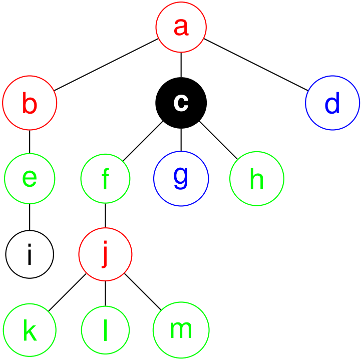
$$\boldsymbol{\alpha} = \left\{ \textcolor{blue}{\bullet}, \textcolor{red}{\bullet}, \textcolor{green}{\bullet}, \bullet \right\} \quad \mathit{ftp}(c) : \Theta \rightarrow \boldsymbol{\alpha} \rightarrow \{0, 1\} \times \mathbb{Z}_3$$



Θ	$\textcolor{blue}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{green}{\bullet}$	$\bullet$
$\theta_{=}$	(0,0)	(0,0)	(0,0)	(1,1)
$\theta_{\downarrow}$	(1,1)	(0,0)	(1,2)	(0,0)
$\textcolor{red}{\theta_{\downarrow\downarrow+}}$	?	?	?	?
$\theta_{\not\sim}$	...	...	...	...
$\theta_{\rightarrow}$	...	...	...	...
...				

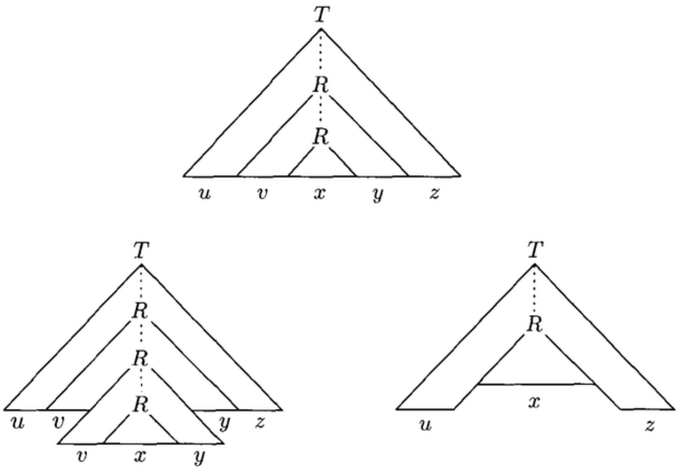
A full type example

$$\boldsymbol{\alpha} = \left\{ \textcolor{blue}{\bullet}, \textcolor{red}{\bullet}, \textcolor{green}{\bullet}, \bullet \right\} \quad \mathit{ftp}(c) : \Theta \to \boldsymbol{\alpha} \to \{0,1\} \times \mathbb{Z}_3$$



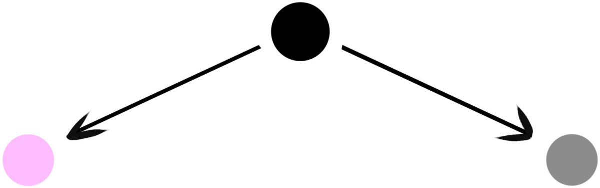
Θ	$\textcolor{blue}{\bullet}$	$\textcolor{red}{\bullet}$	$\textcolor{green}{\bullet}$	$\bullet$
$\theta_{=}$	(0,0)	(0,0)	(0,0)	(1,1)
$\theta_{\downarrow}$	(1,1)	(0,0)	(1,2)	(0,0)
$\theta_{\downarrow\downarrow+}$	(0,0)	(1,1)	(1,0)	(0,0)
$\theta_{\not\sim}$	...	...	...	...
$\theta_{\rightarrow}$	...	...	...	...
...				

Pumping lemma and a small model property

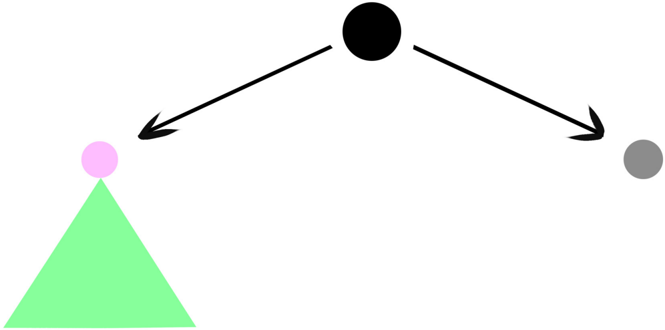


Algorithm.

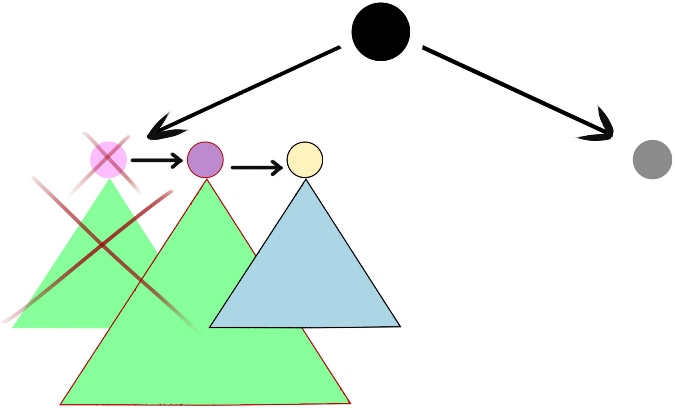
Algorithm.



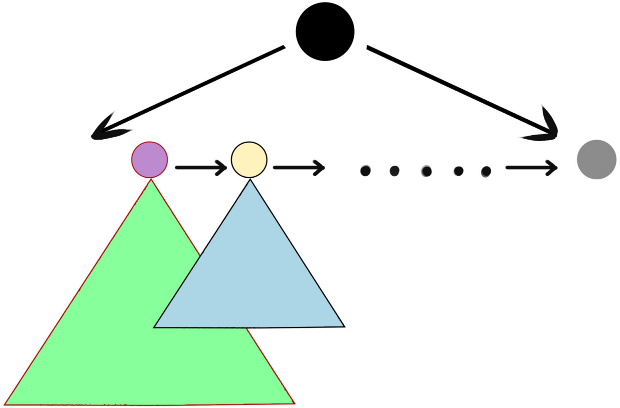
Algorithm.



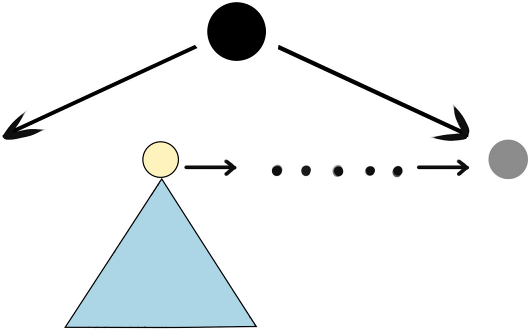
Algorithm.



Algorithm.



Algorithm.



Algorithm.

Procedure 2: Building a subtree rooted at given node

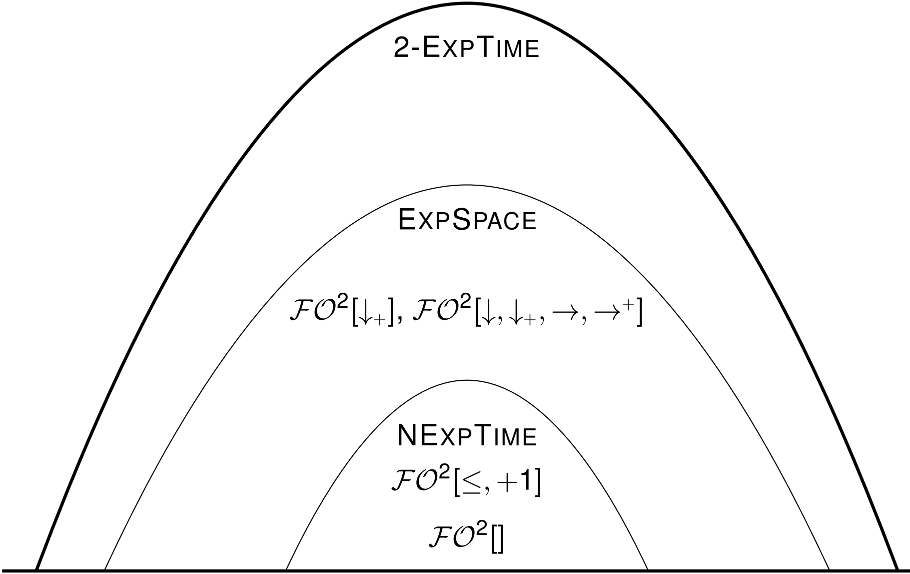
Input: Formula $\varphi \in \mathrm{FO}^2_{\mathrm{MOD}}[\downarrow, \downarrow^+, \rightarrow, \rightarrow^+]$ in normal form, full type $\overline{\alpha}$ of a starting node, and current level $\mathrm{Lvl} \in \mathbb{N}$.

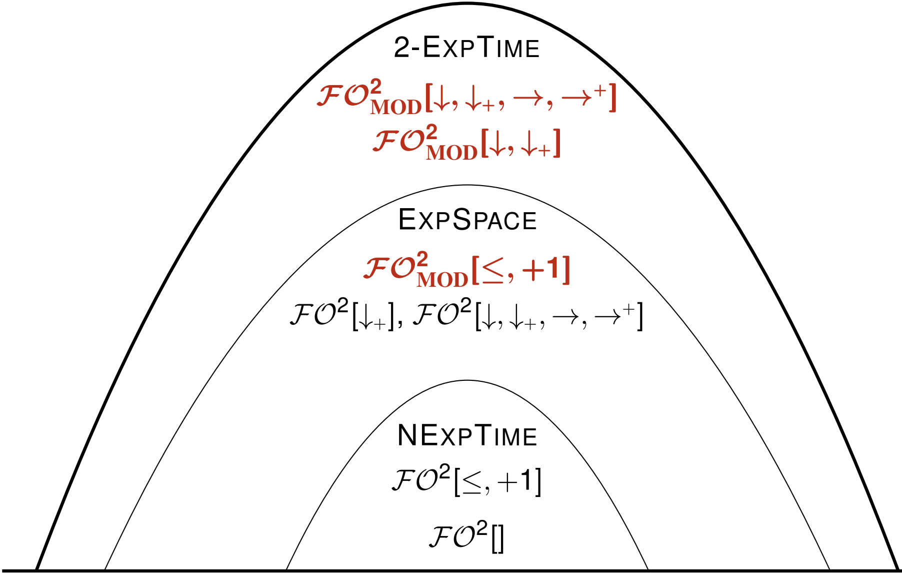
if not is-φ-consistent($\overline{\alpha}$) **then reject** `// See Definition 5`
if $\mathrm{Lvl} \geq \mathfrak{f}(\varphi)$ **then reject** `// Path too long`
if $\overline{\alpha}(\theta_{\downarrow})$ is zero **then accept** `// Last node on the path`
Guess the degree $\mathrm{Deg} \in [1, \mathfrak{f}(\varphi)]$ of a node
Guess the full type $\overline{\beta}$ of the leftmost child and check if its a valid leftmost son of $\overline{\alpha}$
$O_{\theta_{\downarrow}} := \overline{\beta}(\theta_{=})$ `// Types of children guessed so far`
$O_{\theta_{\downarrow\downarrow^+}} := \overline{\beta}(\theta_{\downarrow}) \oplus \overline{\beta}(\theta_{\downarrow\downarrow^+})$ `// Types of descendants guessed so far`
while $Deg > 1$ **do**
    Run in parallel Procedure 2 on $(\varphi, \overline{\beta}, \mathrm{Lvl} + 1)$ `// Alternation here`
    Guess a full type $\overline{\gamma}$ of the right brother of $\overline{\beta}$ and check consistency with $\overline{\alpha}$
    $O_{\theta_{\downarrow}} := O_{\theta_{\downarrow}} \oplus \overline{\gamma}(\theta_{=})$, $O_{\theta_{\downarrow\downarrow^+}} := O_{\theta_{\downarrow\downarrow^+}} \oplus \overline{\gamma}(\theta_{\downarrow}) \oplus \overline{\gamma}(\theta_{\downarrow\downarrow^+})$ `// Updating obligations`
    $\overline{\beta} := \overline{\gamma}$, $\mathrm{Deg} := \mathrm{Deg} - 1$
Run in parallel Procedure 2 on $(\varphi, \overline{\beta}, \mathrm{Lvl} + 1)$ `// Last child`
if $\overline{\beta}(\theta_{\rightarrow})$ is not zero **then reject** `// Not valid last node on →-path.`
if $\overline{\alpha}(\theta_{\downarrow}) = O_{\theta_{\downarrow}}$ and $\overline{\alpha}(\theta_{\downarrow\downarrow^+}) = O_{\theta_{\downarrow\downarrow^+}}$ **then accept else reject**

Conclusions

Open problems

- Establish the complexity of missing subfragments for $\mathcal{FO}^2_{\mathrm{MOD}}$
- Guarded fragment restriction, UAR restriction
- Develop equivalent version of CTL, CLT*, PDL, and so on.
- $\mathcal{FO}^2_{\mathrm{MOD}}$ on arbitrary structures


2-EXPTIME
EXPSPACE
$\mathcal{FO}^2[\downarrow_+]$, $\mathcal{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$
NEXPTIME
$\mathcal{FO}^2[\leq, +1]$
$\mathcal{FO}^2[]$


2-EXPTIME
$\mathcal{FO}^2_{\mathrm{MOD}}[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$
$\mathcal{FO}^2_{\mathrm{MOD}}[\downarrow, \downarrow_+]$
EXPSPACE
$\mathcal{FO}^2_{\mathrm{MOD}}[\leq, +1]$
$\mathcal{FO}^2[\downarrow_+]$, $\mathcal{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$
NEXPTIME
$\mathcal{FO}^2[\leq, +1]$
$\mathcal{FO}^2[]$