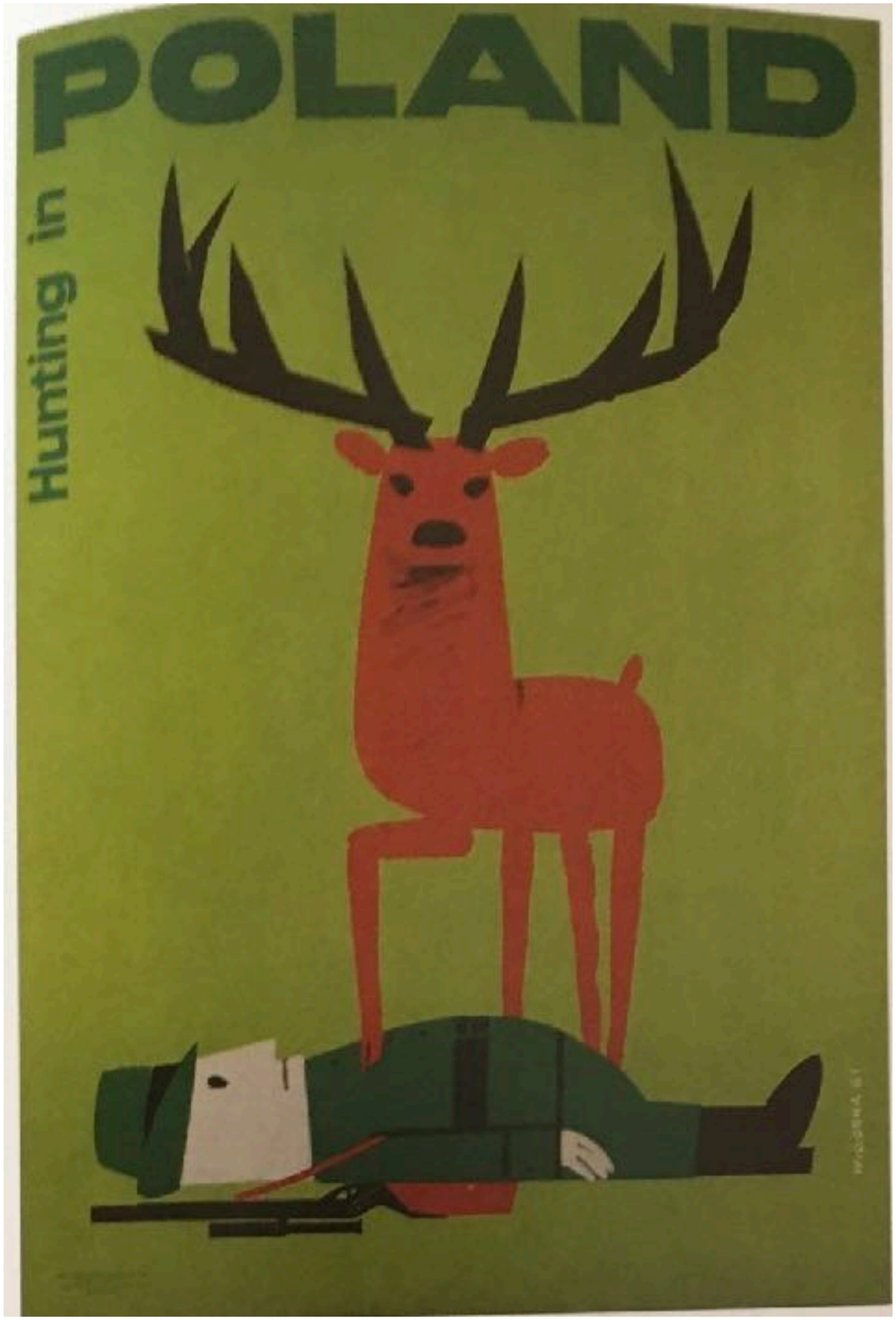
Higher-order MC for small orders



λY-terms without λ-variables

$$a \qquad F \qquad N_1 N_2 \qquad YF.N$$

How Böhm trees are calculated

$$BT((YF.N)K) = BT((N[YF.N/X])K)$$

$$BT(aK_1K_2) = \begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ BT(K_1) & & BT(K_2) \end{tikzcd}$$

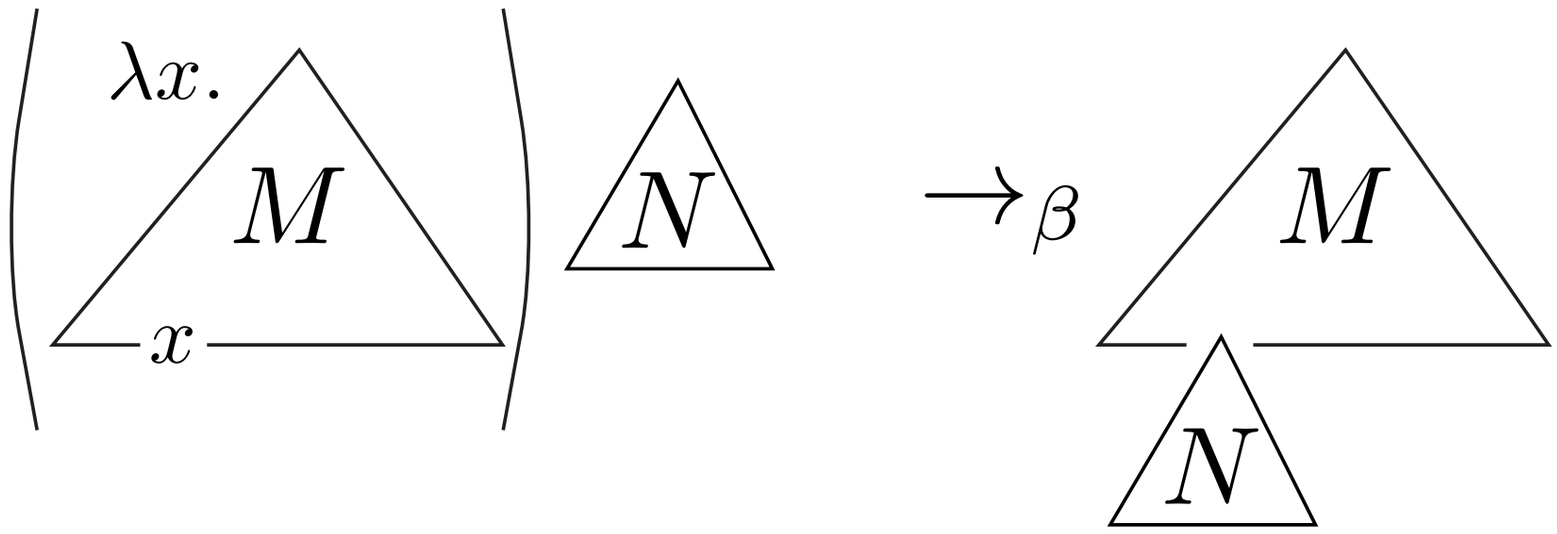
So $BT(M)$ is a regular tree $\equiv$ finite state transition system.

λ-calculus (simply typed)

Types: $\mathrm{o},\ A \to B$ **Terms:** $c,\ x,\ MN,\ \lambda x.M$

Typed terms: $c^A,\ x^A,\ \left(M^{(A\to B)}N^A\right)^B,\ \left(\lambda x^A.M^B\right)^{A\to B}$

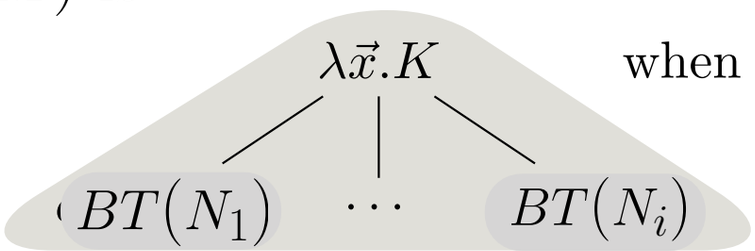
β-reduction: $(\lambda x.M)N \to_\beta M[N/x]$



The unique result is (in some cases) a ranked tree

Tree signature:
All constants of have type of the form $o \to \cdots \to o \to o$, or just o.

$BT(M)$ is

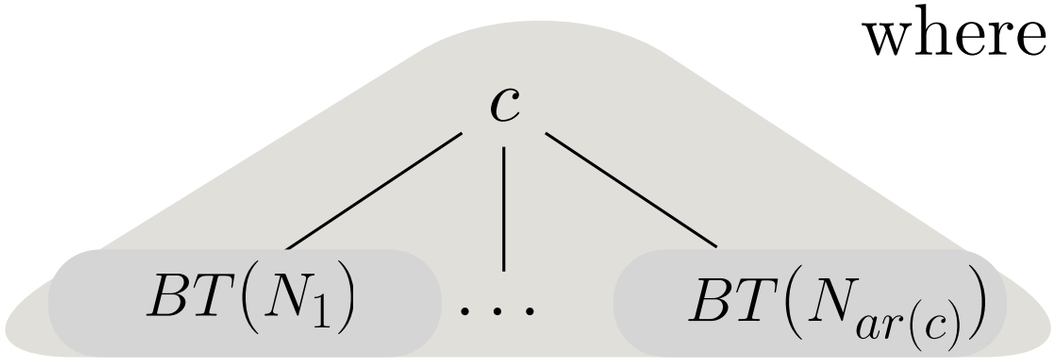


when $M \to_\beta^* \lambda\vec{x}.KN_1 \ldots N_i$

constant or variable

head normal form

where $M \to_\beta^* cN_1 \ldots N_{ar(c)}$



Cor: A normal form of a term $M : 0$ over a tree signature is a finite ranked tree.

Recursive schemes

Hierarchical equations

$$X_n =_\nu \alpha_n(\vec{X})$$
$$\vdots$$
$$X_1 =_\mu \alpha_1(\vec{X})$$

Recursive schemes

$$F_n = \lambda \vec{x}_n . M_n$$
$$\vdots$$
$$F_1 = \lambda \vec{x}_1 . M_1$$

M_i has no λ, is of type o, and contains only variables $\vec{x}_i \cup \{F_1, \ldots, F_n\}$.

Computation rule

$$F_i \vec{N} \to M_i[\vec{N}/x_i]$$

λY-terms with λ-variables of type o

$$a \qquad F \qquad N_1 N_2 \qquad Y\,F.N \qquad x^o \qquad \lambda x^o.M$$

$$\begin{aligned}
\big(NK, \rho, S\big) &\longrightarrow \big(N, \rho, \big(K, \rho\big) \cdot S\big) \\
\big(\lambda x.N, \rho, \big(K, \rho_K\big) \cdot S\big) &\longrightarrow \big(N, \rho\big[x \mapsto \big(K, \rho_K\big)\big], S\big) \\
\big(Y.F.N, \rho, S\big) &\longrightarrow \big(N\big[Y\,F.N/X\big], \rho, S\big) \\
\big(x, \rho, \emptyset\big) &\longrightarrow \big(K, \rho_K, \emptyset\big) \qquad \text{where } \rho\big(x\big) = \big(K, \rho_K\big) \\
\big(a, \rho, \big(K_1, \rho_1\big)\big(K_2, \rho_2\big)\big) &\xrightarrow{a_i} \big(K_i, \rho_i, \emptyset\big)
\end{aligned}$$

λY-terms with λ-variables of type o

$$a \qquad F \qquad N_1 N_2 \qquad YF.N \qquad x^o \qquad \lambda x^o.M$$

Krivine machine

$$\begin{aligned}
(NK, \rho, S) &\longrightarrow (N, \rho, (K, \rho) \cdot S) \\
(\lambda x.N, \rho, (K, \rho_K) \cdot S) &\longrightarrow (N, \rho[x \mapsto (K, \rho_K)], S) \\
(Y.F.N, \rho, S) &\longrightarrow (N[YF.N/X], \rho, S) \\
(x, \rho, \emptyset) &\longrightarrow (K, \rho_K, \emptyset) \qquad \text{where } \rho(x) = (K, \rho_K) \\
(a, \rho, (K_1, \rho_1)(K_2, \rho_2)) &\xrightarrow{a_i} (K_i, \rho_i, \emptyset)
\end{aligned}$$

Example:

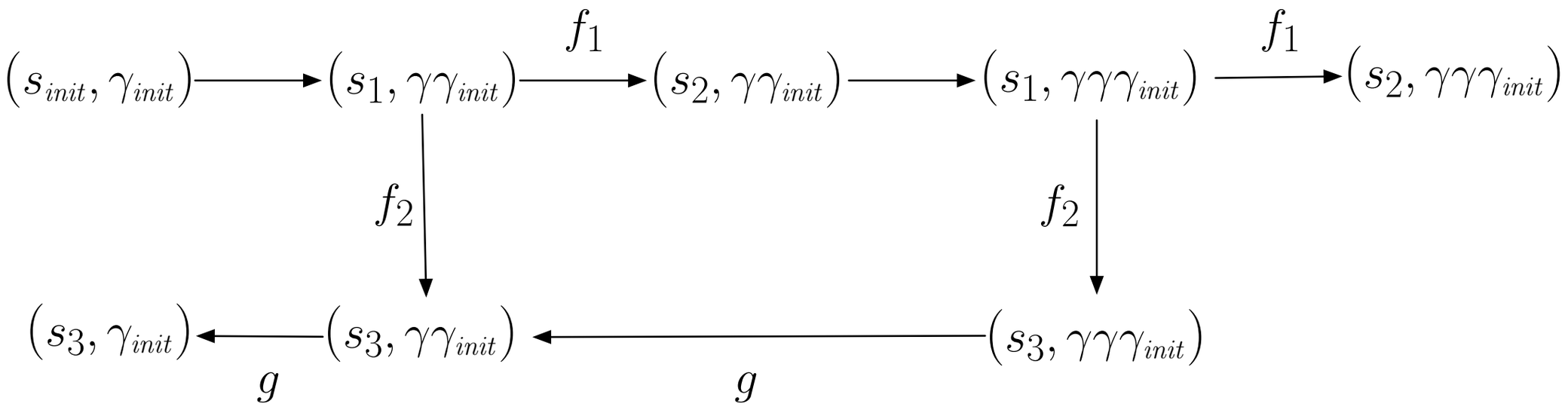
$$(\lambda x.d(dx))c$$

Pushdown automaton

$$\mathcal{P} = \langle S, \Sigma, \Gamma, s_{init}, \bot, \delta \rangle$$

$$\delta(s,\gamma) = \begin{cases} (s', pop) \\ (s', push(\gamma')) \\ f(s_1, \ldots, s_k) \end{cases} \text{ deterministic}$$

$$BT(s,w) = f(BT(s_1,w'), \ldots, BT(s_k,w') \quad \text{if } (s,w) \longrightarrow^* (s',w') \longrightarrow f(s_1,\ldots,s_k)$$
$$= \Omega \text{ otherwise.}$$



λY-terms with λ-variables of type o

$$a \qquad F \qquad N_1 N_2 \qquad Y\,F.N \qquad x^o \qquad \lambda x^o.M$$

$$\begin{aligned}
\big(NK, \rho, S\big) &\longrightarrow \big(N, \rho, \big(K, \rho\big) \cdot S\big) \\
\big(\lambda x.N, \rho, \big(K, \rho_K\big) \cdot S\big) &\longrightarrow \big(N, \rho\big[x \mapsto \big(K, \rho_K\big)\big], S\big) \\
\big(Y.F.N, \rho, S\big) &\longrightarrow \big(N\big[YF.N/X\big], \rho, S\big) \\
\big(x, \rho, \emptyset\big) &\longrightarrow \big(K, \rho_K, \emptyset\big) \qquad \text{where } \rho(x) = (K, \rho_K)
\end{aligned}$$

$$\big(a, \rho, \big(K_1, \rho_1\big)\big(K_2, \rho_2\big)\big) \xrightarrow{a_i} \big(K_i, \rho_i, \emptyset\big)$$

Krivine machine can be simulated by a pushdown

Simulating evaluation with a pushdown

Pushdown alphabet: (var,term)

Special variables $\zeta_1, \ldots, \zeta_l$ for closures on the stack

Look up function $\mathit{find}(z, w)$

$$\mathit{find}(z,w) = \begin{cases} (N, \mathit{pop}(w)) & \text{if } \mathit{top}(w) = (z, N) \\ \mathit{find}(\zeta_i, w) & \text{if } \mathit{top}(w) = (z, \zeta_i) \\ \mathit{find}(z, \mathit{pop}(w)) & \text{otherwise} \end{cases}$$

Def: $\rho[w](x) = (N, \rho[w'])$ when $\mathit{find}(x, w) = (N, w')$.

Invariant:

KM	pushdown
$(N, \rho, (K_1, \rho_1) \ldots (K_l, \rho_l))$	w

$$\rho(x) = \rho[w](x) \qquad (K_i, \rho_i) = \rho[w](\zeta_i)$$

Def: $\rho[w](x) = (N, \rho[w'])$ when $\mathit{find}(x, w) = (N, w')$.

Invariant:

KM	pushdown
$(N, \rho, (K_1, \rho_1) \ldots (K_l, \rho_l))$	(N, w)
$\rho(x) = \rho[w](x)$	$(K_i, \rho_i) = \rho[w](\zeta_i)$

$$
\begin{array}{rll}
(NK, \rho, S) & \longrightarrow (N, \rho, (K, \rho) \cdot S) & (N, \mathit{push}(\zeta_{|S|+1}, K)) \\
(\lambda x.N, \rho, (K, \rho_K) \cdot S) & \longrightarrow (N, \rho[x \mapsto (K, \rho_K)], S) & (N, \mathit{push}(x, \zeta_{|S|+1})) \\
(Y.F.N, \rho, S) & \longrightarrow (N[YF.N/X], \rho, S) & \\
(x, \rho, \emptyset) & \longrightarrow (K, \rho_K, \emptyset) & \mathit{find}(x) \\
(a, \rho, (K_1, \rho_1)(K_2, \rho_2)) & \xrightarrow{a_i} (K_i, \rho_i, \emptyset) & a(\mathit{find}(\zeta_1), \mathit{find}(\zeta_2))
\end{array}
$$

Increasing the order

λY-terms with λ-variables of type o

$$a \qquad F \qquad N_1 N_2 \qquad YF.N \qquad x^o \qquad \lambda x^o.M$$

λY-terms with λ-variables of type of order 1, like o->o

$$a \qquad F \qquad N_1 N_2 \qquad YF.N \qquad x^o \qquad \lambda x^o.M \qquad f^{o\to o} \qquad \lambda f^{o\to o}.M$$

Example: $\big(\lambda f.\lambda x.\ f\big(fx\big)\big)ad$

Example: $(\lambda f.\lambda x.\ f(fx))ad$

$$
\begin{aligned}
(NK, \rho, S) &\longrightarrow (N, \rho, (K, \rho) \cdot S) \\
(\lambda x.N, \rho, (K, \rho_K) \cdot S) &\longrightarrow (N, \rho[x \mapsto (K, \rho_K)], S) \\
(Y.F.N, \rho, S) &\longrightarrow (N[YF.N/X], \rho, S) \\
(x, \rho, S) &\longrightarrow (K, \rho_K, S) \qquad \text{where } \rho(x) = (K, \rho_K)
\end{aligned}
$$

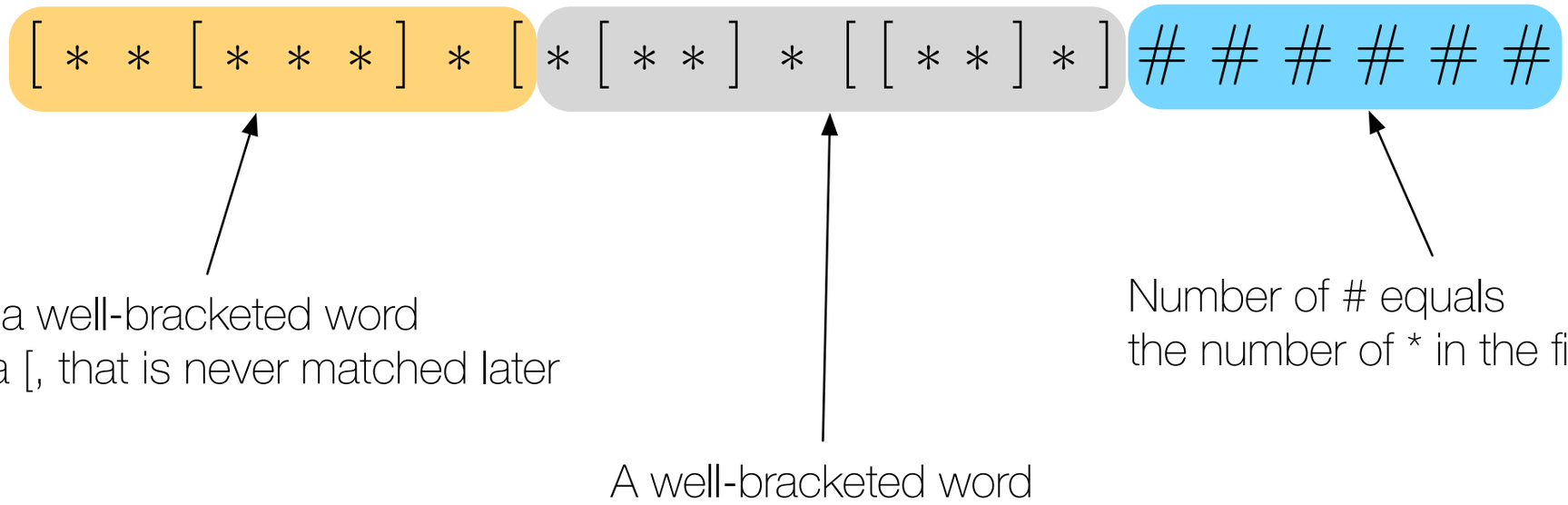
$$
(a, \rho, (K_1, \rho_1)(K_2, \rho_2)) \xrightarrow{a_i} (K_i, \rho_i, \emptyset)
$$

When collapse is needed, Urzyczyn's language

Example from Pawel Parys

Σ consists of $[$, $]$, $*$, $\#$

Urzyczyn's language contains words of the form:



Thm [Parys]**:**

Urzyczyn's language cannot be recognized by a higher-order pushdown automaton (without collapse).

Back to λY-terms with λ-variables of type o

$$a \qquad F \qquad N_1 N_2 \qquad YF.N \qquad x^o \qquad \lambda x^o.M$$

Encoding pushdown automata in λY-terms

$$\mathcal{P} = \langle S, \Sigma, \Gamma, s_{init}, \bot, \delta \rangle$$

Encoding of the stack

$$\begin{aligned} \langle \bot \rangle &= (\bot, \ldots, \bot) \\ \langle aw \rangle &= (F^a_{s_1}\langle w \rangle, \ldots, F^a_{s_n}\langle w \rangle) \end{aligned}$$

Encoding of the automaton

$$\mathcal{I} = F^{\bot}_{s_{init}}\langle \bot \rangle$$

$$\begin{aligned} F^a_s &= \lambda \vec{x}.\ x^{s'} & \delta(s,a) &= (s', pop) \\ F^b_t &= \lambda \vec{x}.\ F^c_{t'}(F^b_{s_1}\vec{x}) \ldots (F^b_{s_n}\vec{x}) & \delta(t,b) &= (t', push(c)) \\ F^a_s &= f(F^b_{t_1}\vec{x}) \ldots (F^b_{t_k}\vec{x}) & \delta(s,a) &= f(t_1, \ldots, t_k) \end{aligned}$$

Invariant: If $(s, av) \longrightarrow (t, bw)$ then $F^a_s\langle v \rangle \longrightarrow F^b_t\langle w \rangle$.

Cor: Deterministic pushdown automata and λY-terms with λ-variables of type o define the same class of trees. [Courcelle'76]

Model checking λY-terms with vars of type o

Pushdown automaton

$$PG = \langle S_E, S_A, \Sigma, \Gamma, s_{init}, \bot, \delta, \Omega, mem \rangle$$

Parity automaton

$$\mathcal{A} = \langle Q,\ \Sigma_{ar},\ q_0,\ \delta : Q \times \Sigma_{ar} \to \mathcal{P}(\mathcal{P}(Q)),\ \Omega : Q \to [d] \rangle$$

Pushdown game

$$PG = \langle S_E, S_A, \Sigma, \Gamma, q_{init}, \bot, \delta, \Omega \rangle \qquad \delta(s,\gamma) = \begin{cases} (s', pop) \\ (s', push(\gamma')) \\ f(s_1, \ldots, s_k) \end{cases}$$

$(s,w) \to (s',w')$ unique successor according to δ

$(s,w) \xrightarrow{f} (s_1,w) \ldots (s_k,w)$ Eve chooses iff $s \in S_E$.

Solving pushdown games: Decide if Eve wins from (s_{init}, q_{init}) in PG.

Making pushdown games rank aware

$$PG = \langle S_E, S_A, \Sigma, \Gamma, s_{init}, \bot, \delta, \Omega, mem \rangle$$

Rank aware game: there is a function $mem : s \to [d]$ s.t.
if $(s_1, w) \longrightarrow (s_2, \gamma w) \longrightarrow^* (s_2, \gamma w)$ then $mem(s_3) =$ "max rank on the path".

Solving pushdown games: Decide if Eve wins from (s_{init}, q_{init}) in a rank aware PG.

Reduction to a finite game

$$PG = \langle S_E, S_A, \Sigma, \Gamma, s_{init}, \bot, \delta, \Omega, mem \rangle$$

Finite game FG:

Positions: (s, γ, R) where $R \subseteq S$

if $\delta(s, \gamma) = (s', push(\gamma'))$.

$$(s,\gamma,R) \longrightarrow [R', s, \gamma, R] \xrightarrow{mem(s_1)} (s_1, \gamma, R)$$
$$[R', s, \gamma, R] \xrightarrow{mem(s_k)} (s_k, \gamma, R)$$
$$[R', s, \gamma, R] \longrightarrow (s', \gamma', R')$$

$$R' = \{s_1, \ldots, s_k\}$$

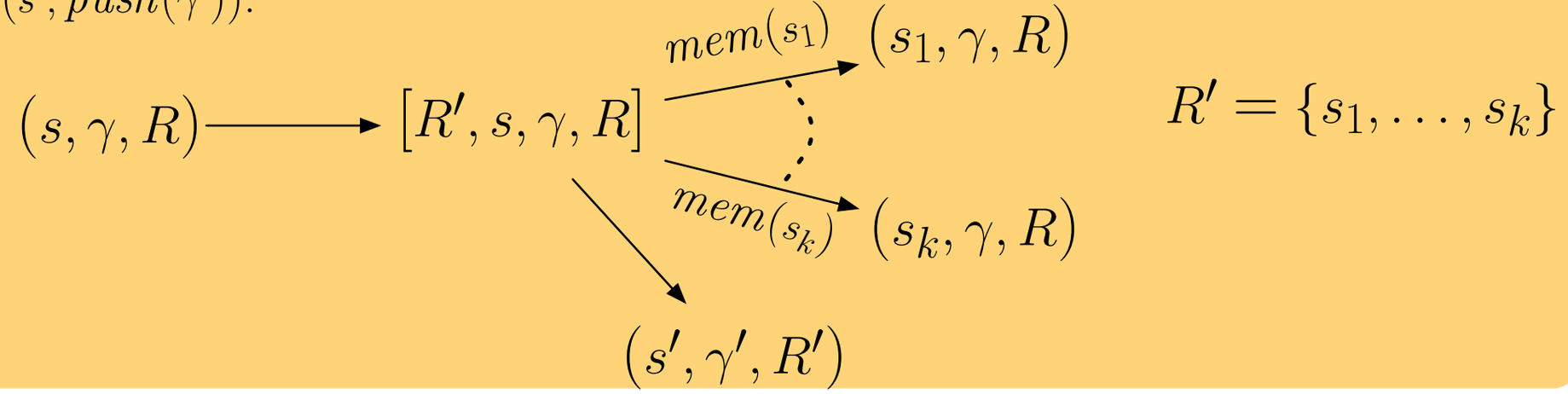
if $\delta(s, \gamma) = (s', pop)$

$$(s,\gamma,R) \longrightarrow \top \quad \text{when } s' \in R$$
$$(s,\gamma,R) \longrightarrow \bot \quad \text{when } s' \notin R$$

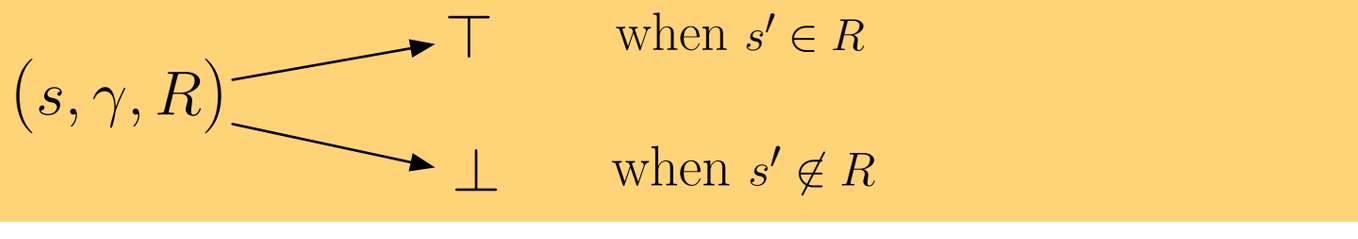
if $\delta(s, \gamma) = f(s_1, \ldots, s_k)$

$$(s,\gamma,R) \longrightarrow \langle s, (s_1, \ldots, s_k), \gamma, R \rangle \longrightarrow (s_1, \gamma, R)$$
$$\langle s, (s_1, \ldots, s_k), \gamma, R \rangle \longrightarrow (s_k, \gamma, R)$$

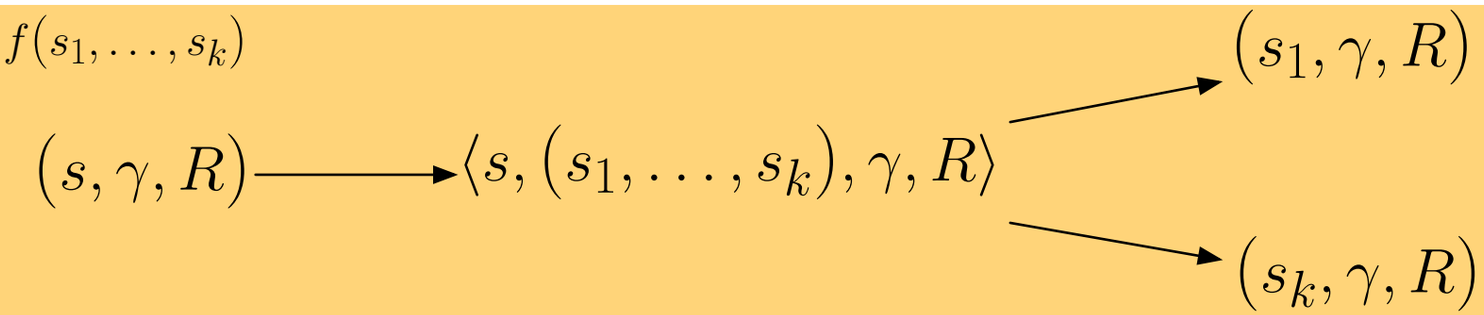
if $\delta(s,\gamma) = (s', push(\gamma'))$.



if $\delta(s,\gamma) = (s', pop)$

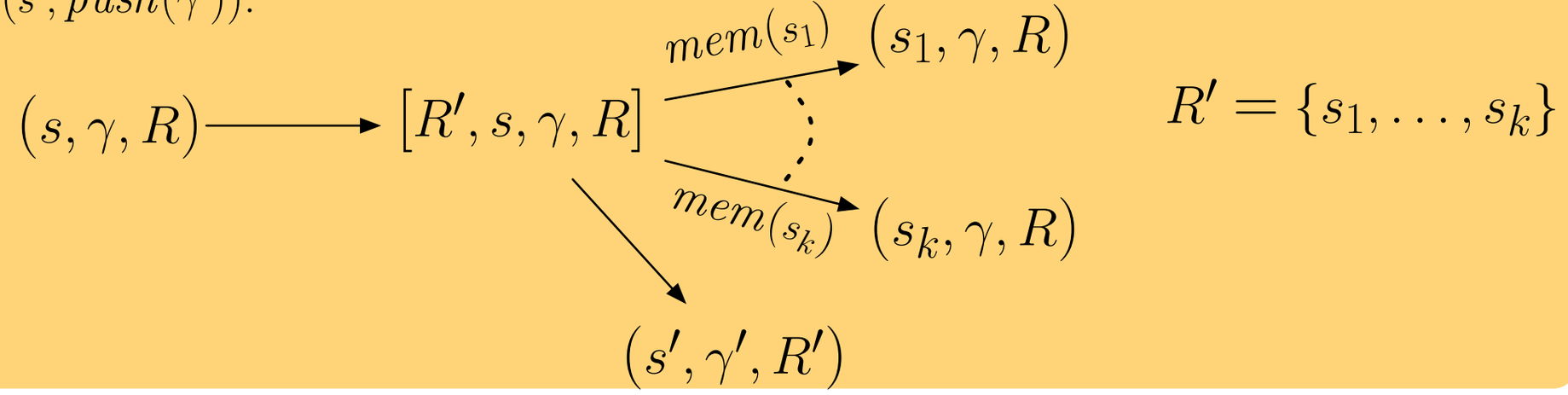


if $\delta(s,\gamma) = f(s_1,\ldots,s_k)$



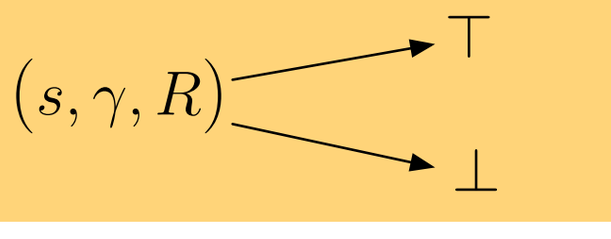
Thm: Eve wins from $(s_{init}, \bot)$ in PG iff she wins from $(s_{init}, \bot, \emptyset)$ in FG.

if $\delta(s,\gamma) = (s', push(\gamma'))$.

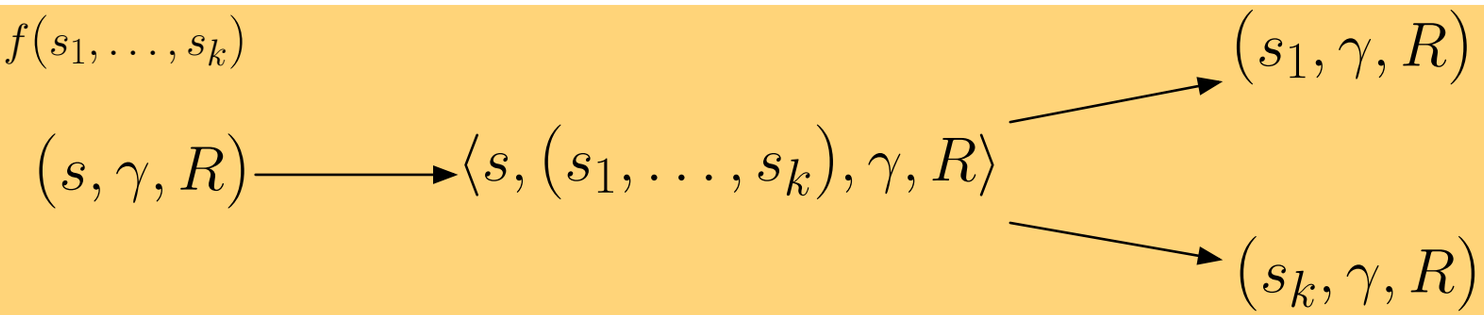


$R' = \{s_1,\ldots,s_k\}$

if $\delta(s,\gamma) = (s', pop)$



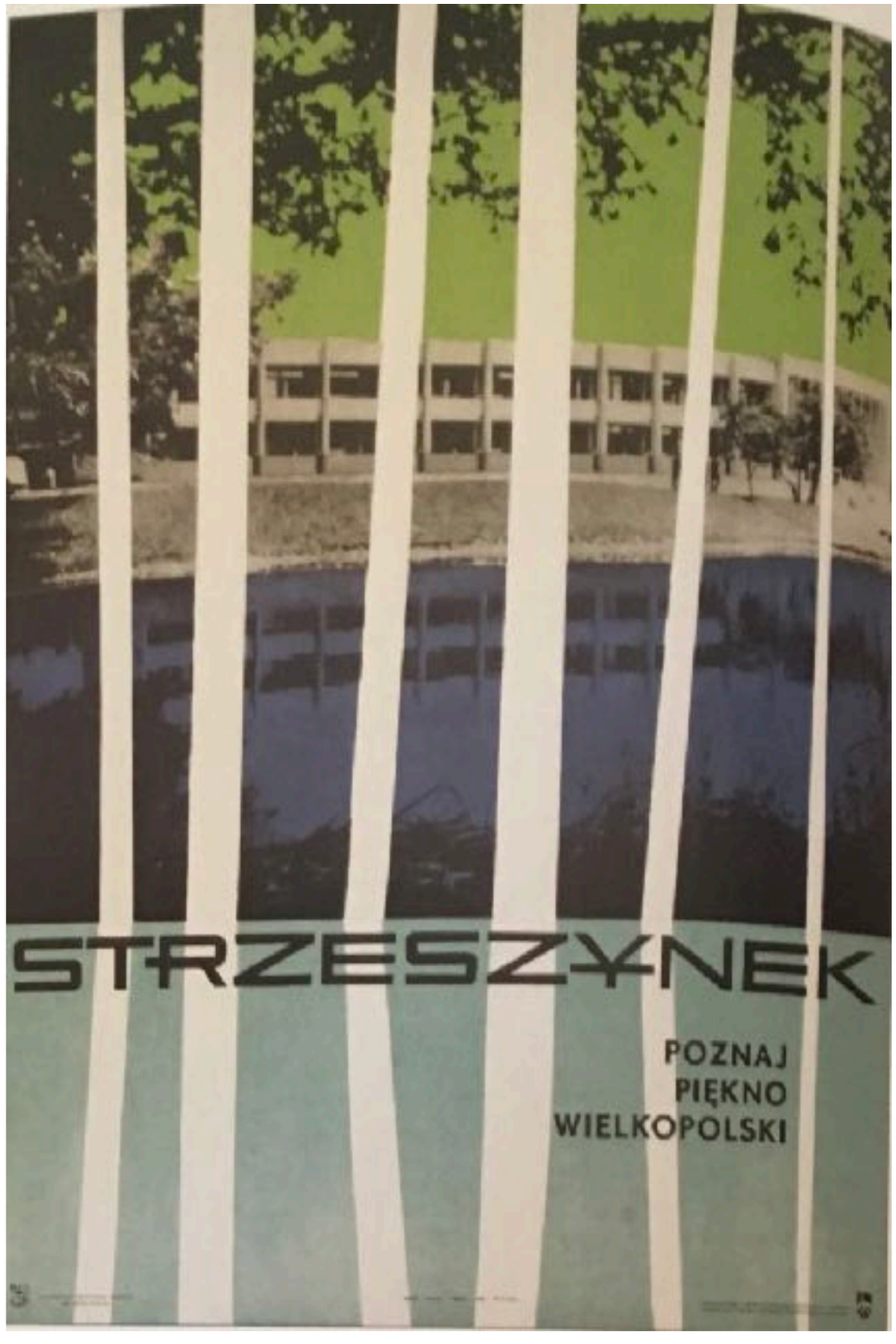
if $\delta(s,\gamma) = f(s_1,\ldots,s_k)$



Thm: Eve wins from $(s_{init},\bot)$ in PG iff she wins from $(s_{init},\bot,\emptyset)$ in FG.

Cor: It is decidable who wins in a given pushdown game.

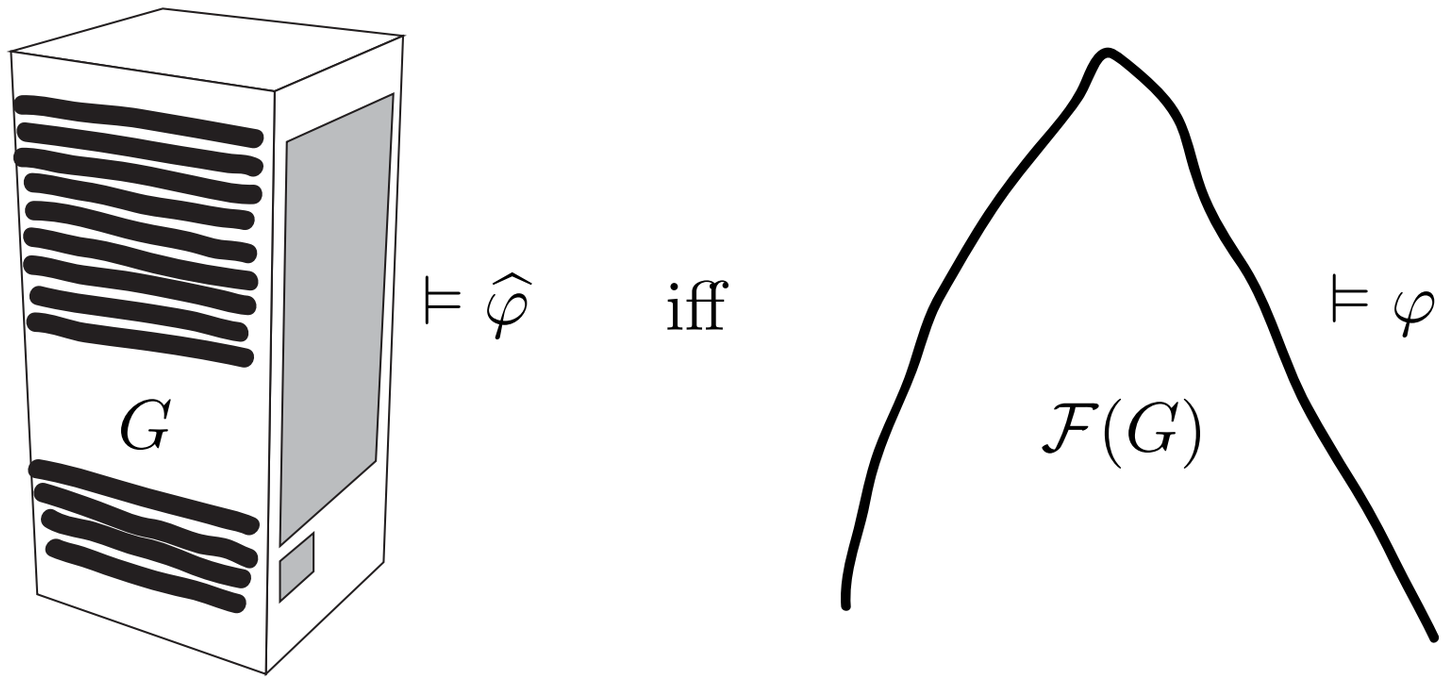
What are Böhm trees of λY-terms?



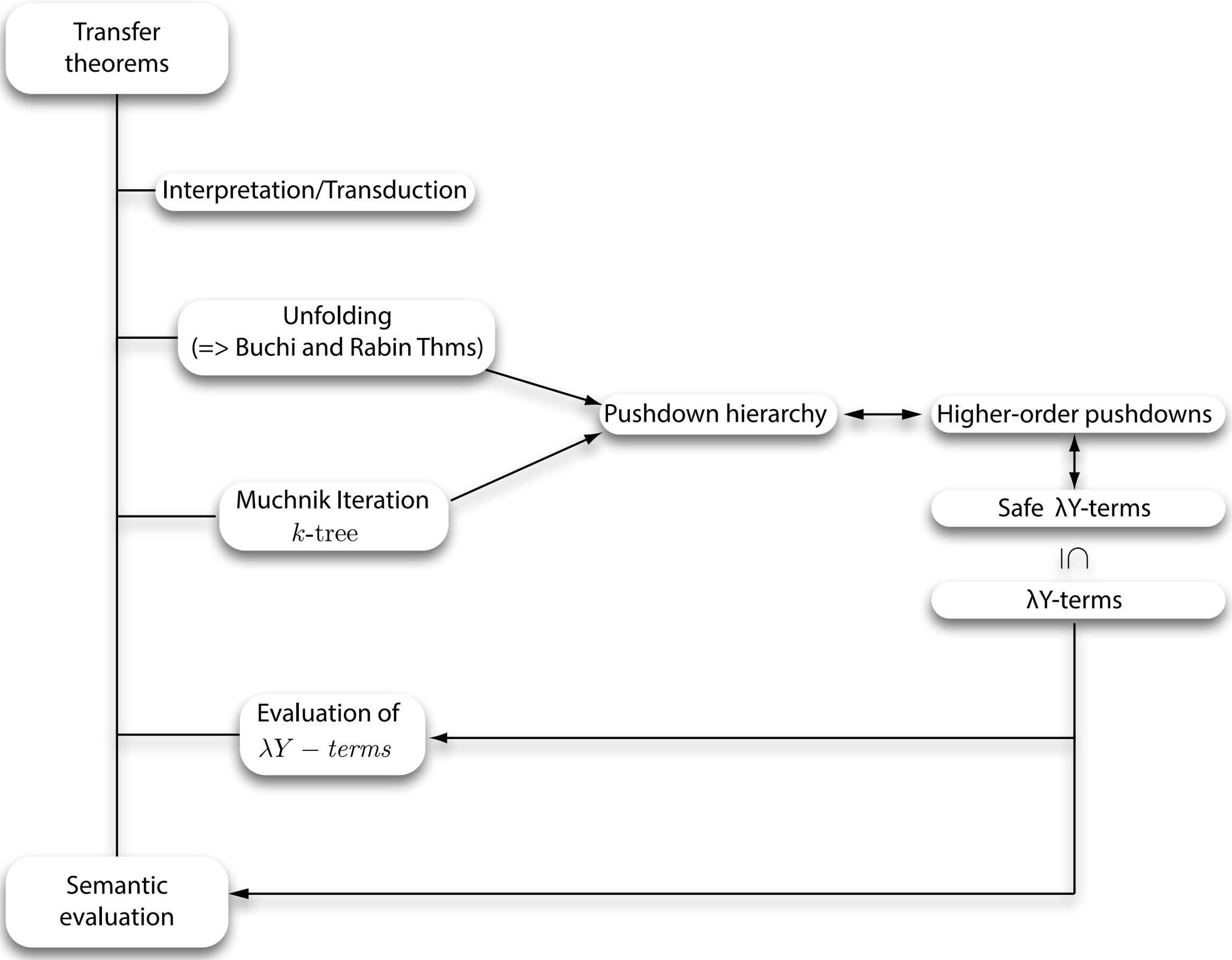
Consider an operation $\mathcal{F}$ on graphs.

Transfer property for $\mathcal{F}$
For every φ one can effectively construct $\widehat{\varphi}$, s.t., for every G:

$$G \vDash \widehat{\varphi} \qquad \text{iff} \qquad \mathcal{F}(G) \vDash \varphi.$$



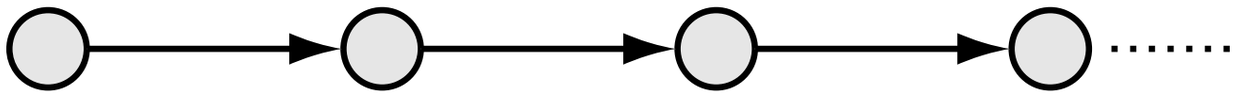
We say in this case that $\mathcal{F}$ is **MSO-compatible**.


Transfer theorems
Interpretation/Transduction
Unfolding (=> Buchi and Rabin Thms)
Muchnik Iteration k-tree
Pushdown hierarchy
Higher-order pushdowns
Safe λY-terms
⋂
λY-terms
Evaluation of $\lambda Y - terms$
Semantic evaluation

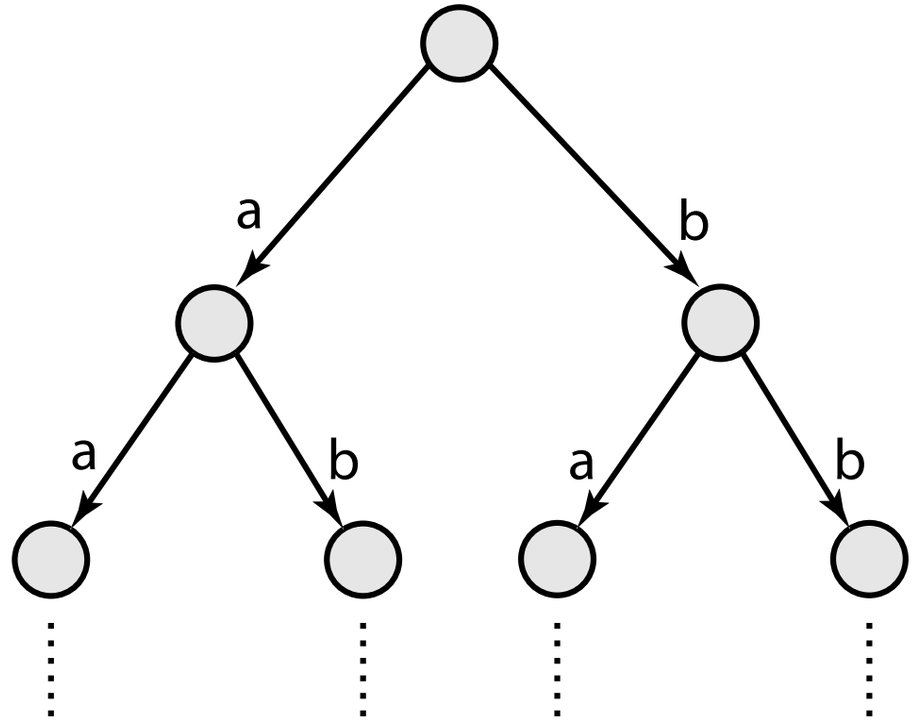
Transfer theorems

Interpretation/Transduction

Infinite sequence:



Infinite full binary tree:

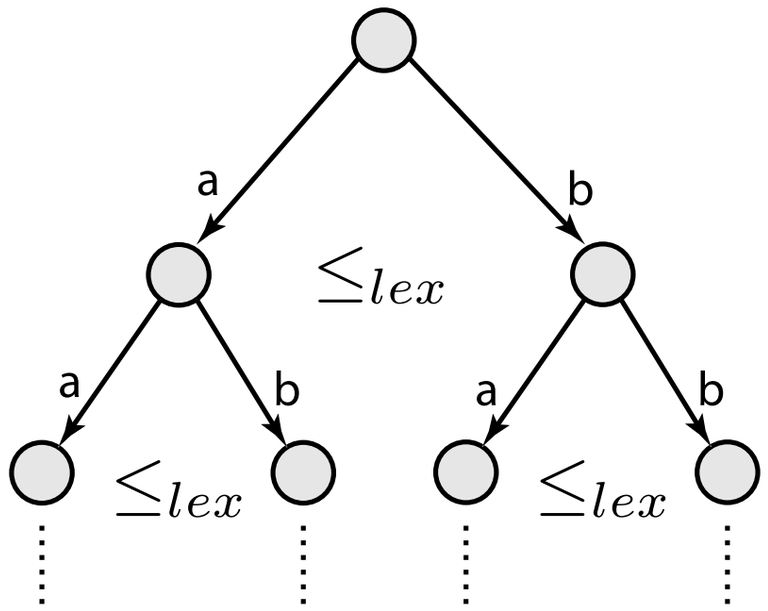


Thm [Buchi'62]**:**
MSOL theory of the infinite sequence is decidable.

Thm [Rabin'69]**:**
MSOL theory of the infinite binary tree is decidable.

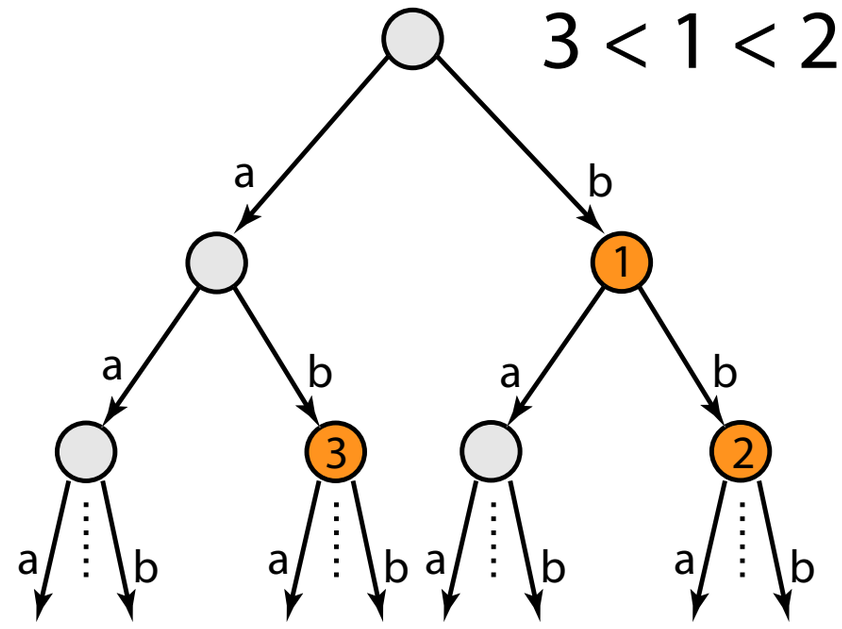
Example:
every countable order is interpretable in a tree

Lexicographic order in a tree



$\leq_{lex}$ is MSOL definable.

A countable ordering can be injected in the lex-order



Cor: MSOL theory of countable orderings is decidable.

A MSOL formula φ is true in all countable orders iff

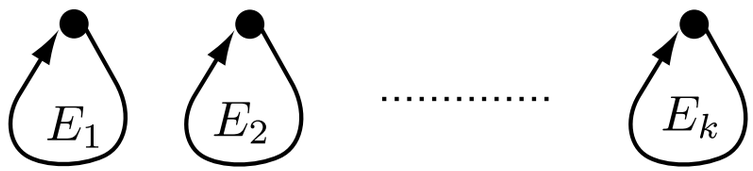
$$\forall X.\ \widehat{\varphi}(X) \text{ is true in the binary tree.}$$

($\widehat{\varphi}(X)$ is obtained by relativising quantification to X, and using $\leq_{lex}$ instead of $\leq$)

MSOL transduction

+ **MSOL interpretation**: using relations defined by formulas

+ **Copying**: duplicating a structure a fixed number of times.



Thm [Courcelle]:
Every MSOL transduction is MSOL compatible.

Many other theories are interpretable in the MSOL theory of the binary tree

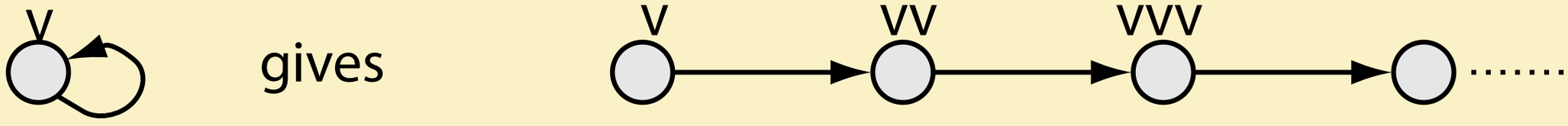
- Countable well-orders
- Presburger arithmetic: <N,+>
- Skolem arithmetic: <N,×>
- Every tree-automatic structure

Transfer theorems

- Interpretation/Transduction
- Unfolding (=> Buchi and Rabin Thms)

Unfolding of a graph form a given node:

the tree of all the paths from the node



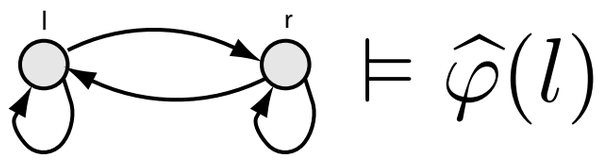
Thm [Courcelle, W.'95]: Unfolding is MSOL compatible.

For every $\varphi(x)$ there is (effectively) $\widehat{\varphi}(x)$ such that
for every graph G and its vertex v_0:

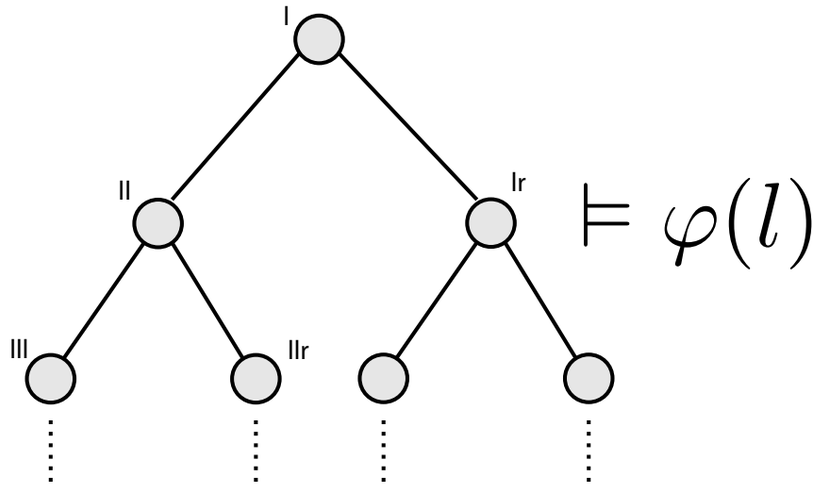
$$G \models \widehat{\varphi}(v_0) \qquad \text{iff} \qquad Unf(G) \models \varphi(v_0)$$

Remark 1: Unfolding cannot be defined by a transduction.

Remark 2: MSO-compatibility of the unfolding implies theorems of Buchi and Rabin.



$\models \widehat{\varphi}(l)$ iff

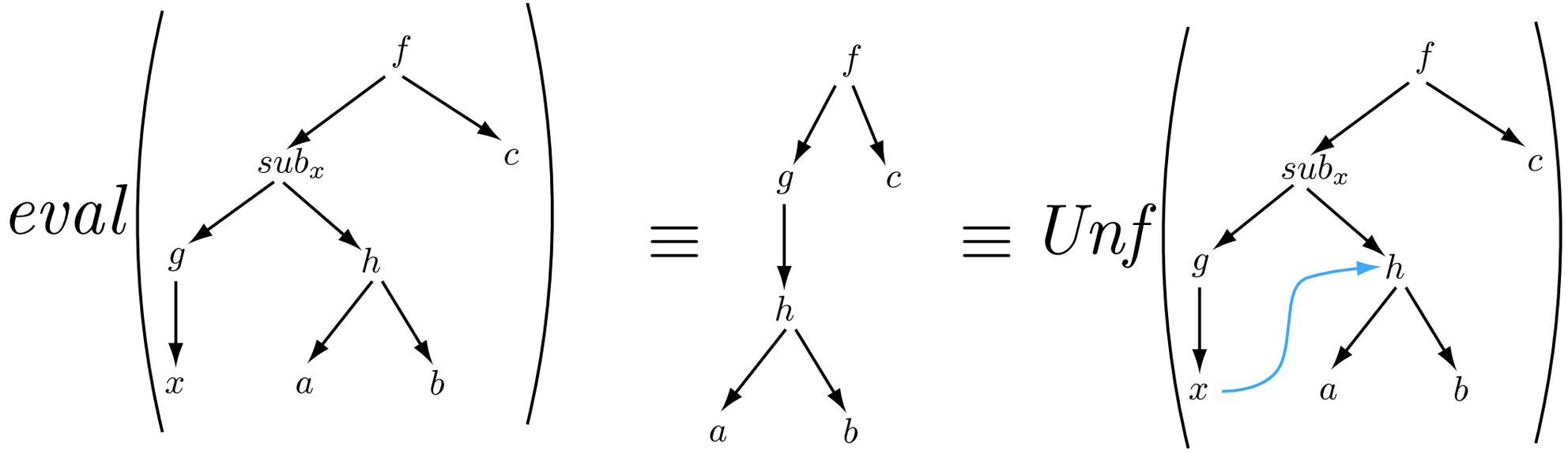


$\models \varphi(l)$

Evaluation of substitutions

Tree with substitutions: function symbols $a, f, g, \ldots$; variables $x, y, \ldots$; and explicit substitutions Sub_x.

$$eval(Sub_x(s,t)) = s[t/x]$$



Thm [Courcelle, Knapik'02]: For a fixed finite set of variables, *eval* on trees with substitutions is MSOL compatible.

Transfer theorems

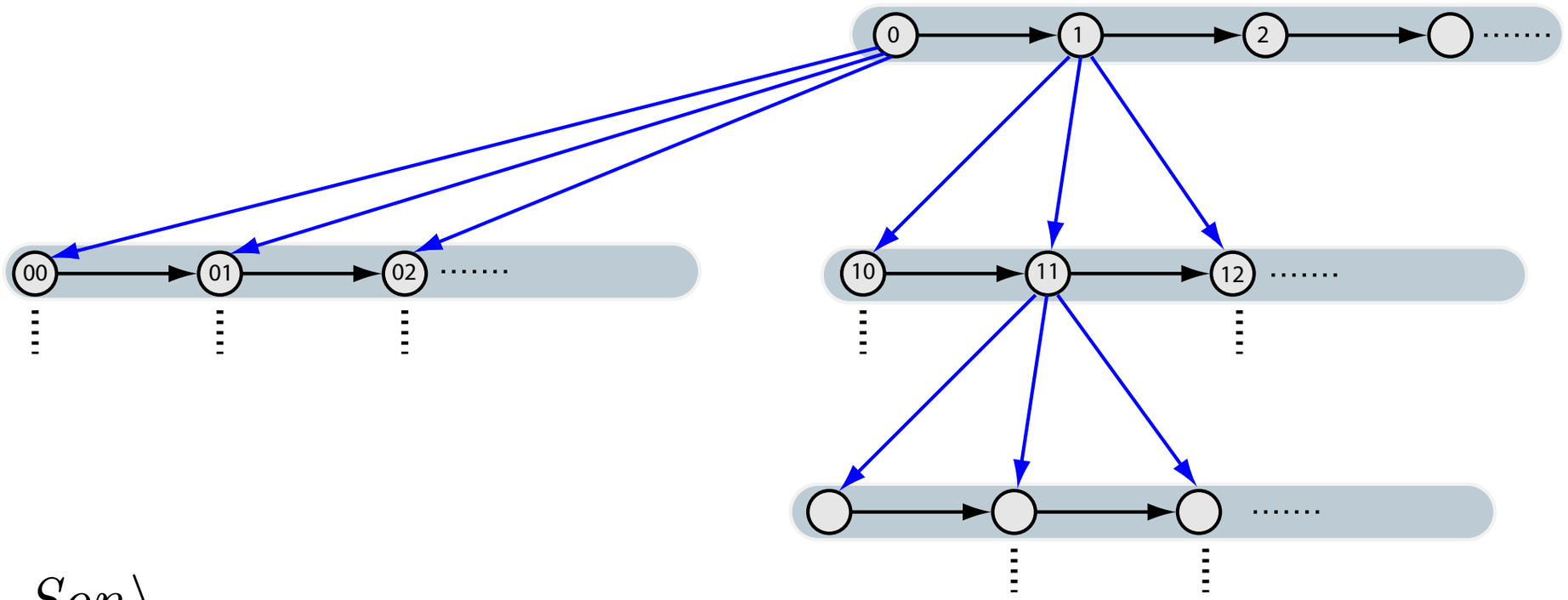
Interpretation/Transduction

Unfolding
(=> Buchi and Rabin Thms)

Muchnik Iteration
k-tree

Stupp's iteration '75

$St($ $)$ is



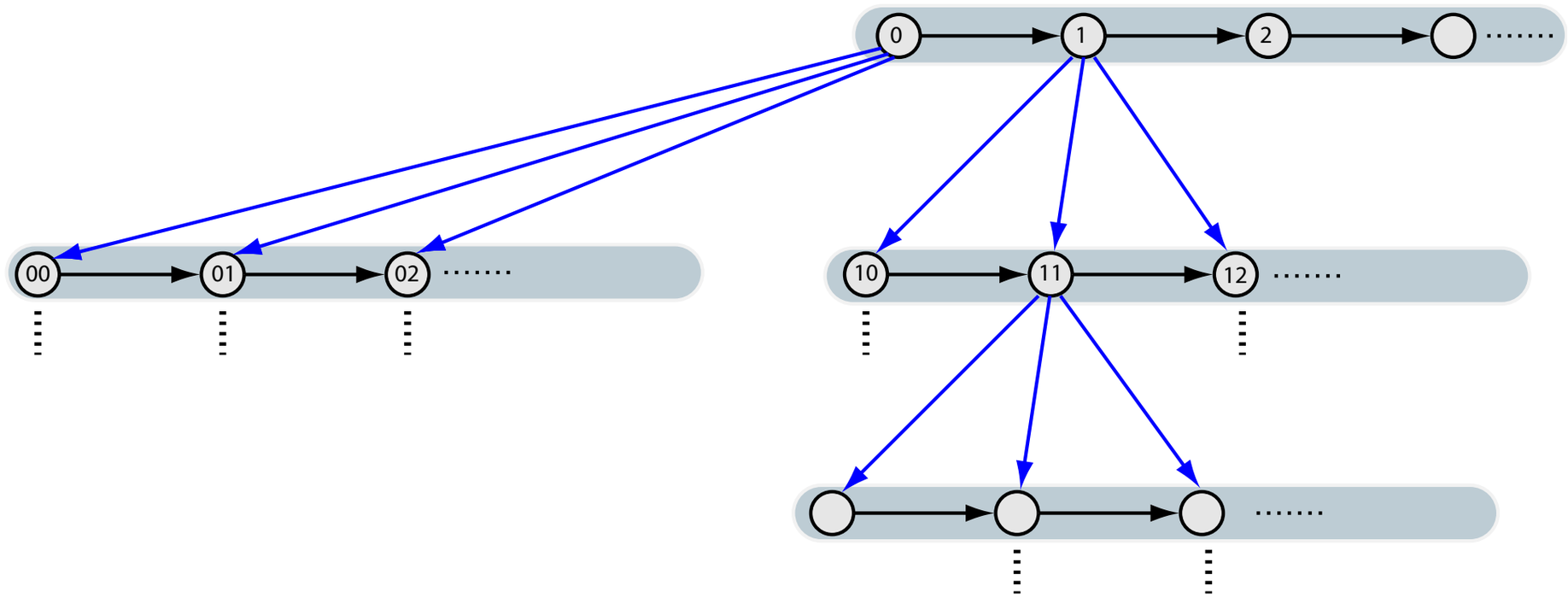
$St(G) = \langle V^+, E^*, Son \rangle$

where for $\mathbf{w} \in V^*$, $u, v \in V$:

- $Son(\mathbf{w}, \mathbf{w}v)$,
- $E^*(\mathbf{w}u, \mathbf{w}v)$ when $E(u, v)$.

Thm [Stupp]: Stupp's iteration is MSOL compatible.

$St($ $)$ is

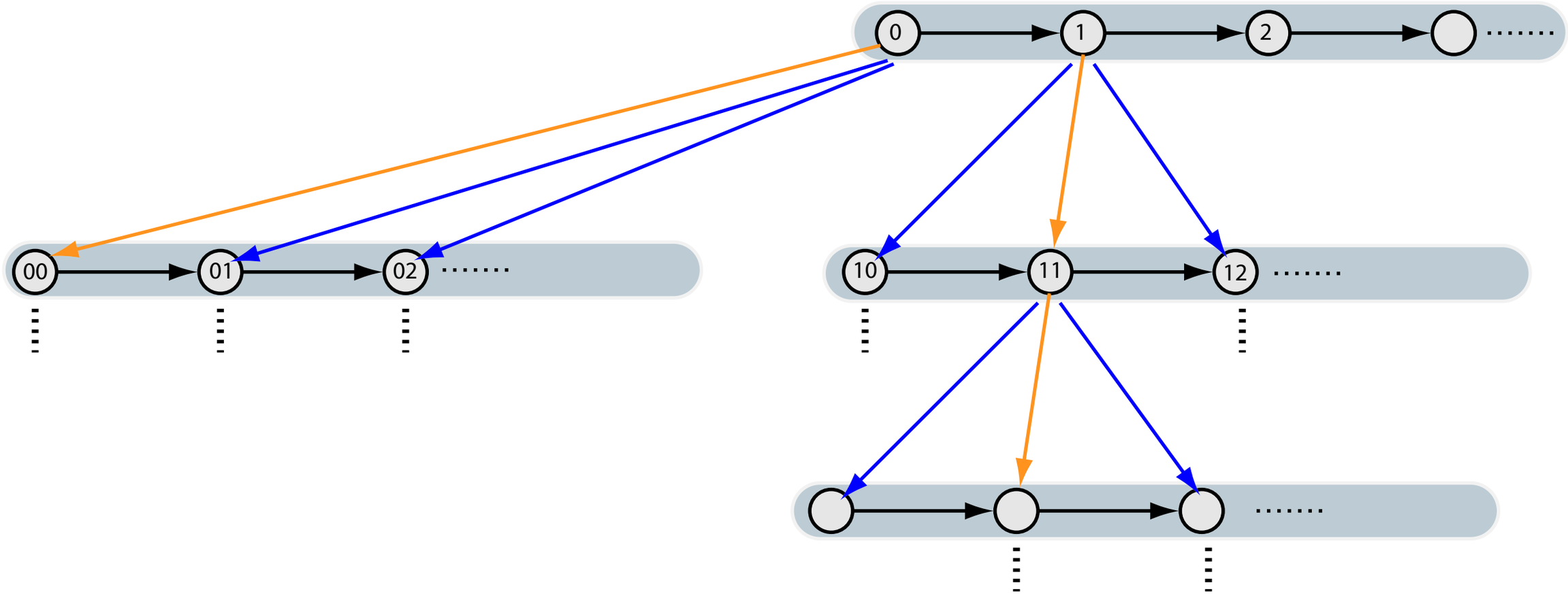


Remark 1: Stupp's iteration of a two node graph gives the full binary tree

Remark 2: Unfolding of a graph may not be definable in the Stupp's iteration of the graph.

Remark 3: Stupp's iteration of the full binary tree is MSO-interpretable in the full binary tree.

Muchnik's iteration '84

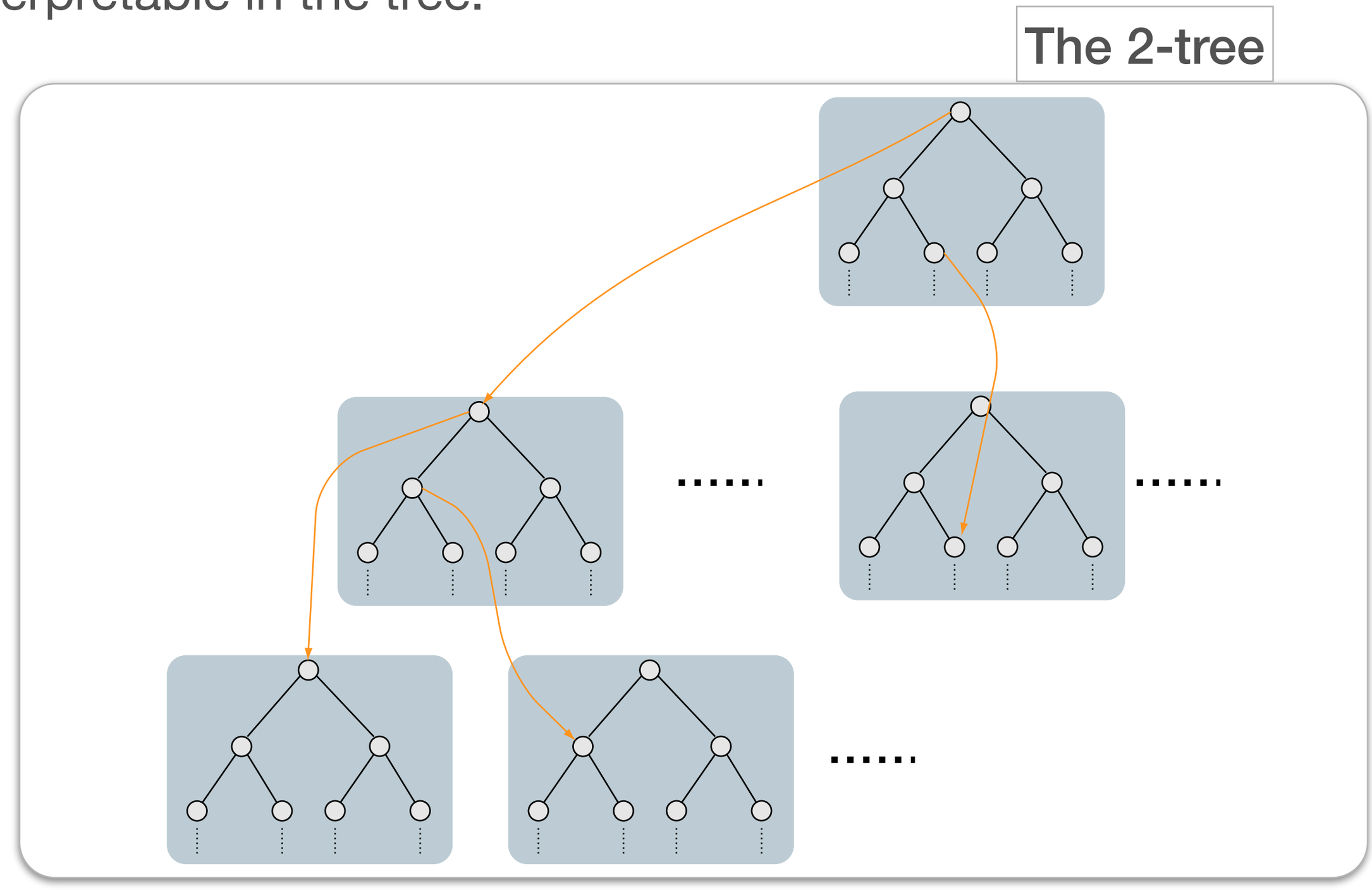


$$G^+ = \langle V^+, E^*, E_{\#}, Son \rangle$$

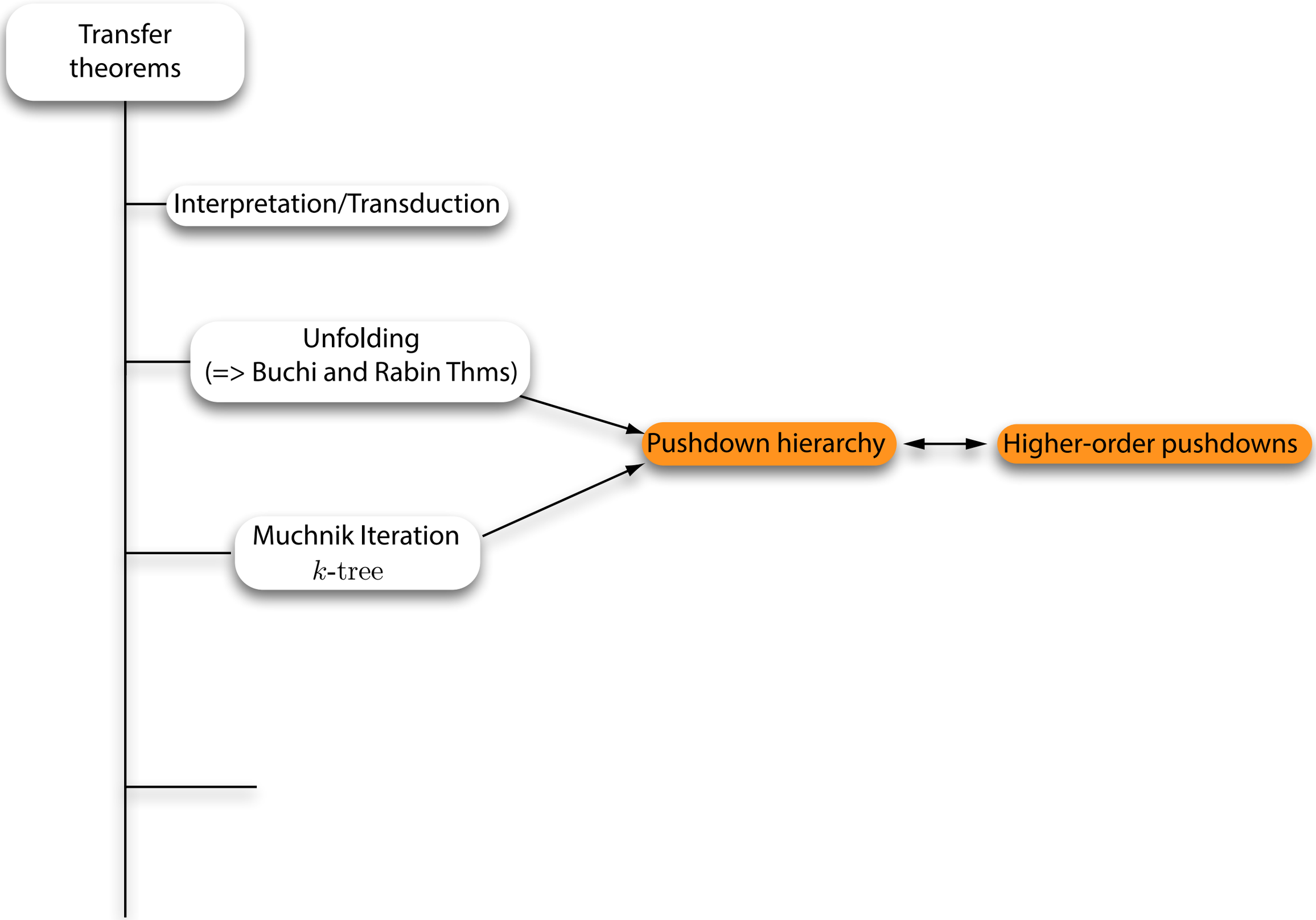
where $E_{\#}(wu, wuu)$ for $w \in V^*$ and $u \in V$.

Thm [Muchnik '84, W. '96]: Muchnik's iteration is MSOL compatible.

Remark 1: Muchnik's iteration of a tree is not MSOL interpretable in the tree.



Remark 2: Unfolding of a graph is definable in the Muchnik's iteration of the graph.


Transfer theorems
Interpretation/Transduction
Unfolding (=> Buchi and Rabin Thms)
Muchnik Iteration
k-tree
Pushdown hierarchy
Higher-order pushdowns

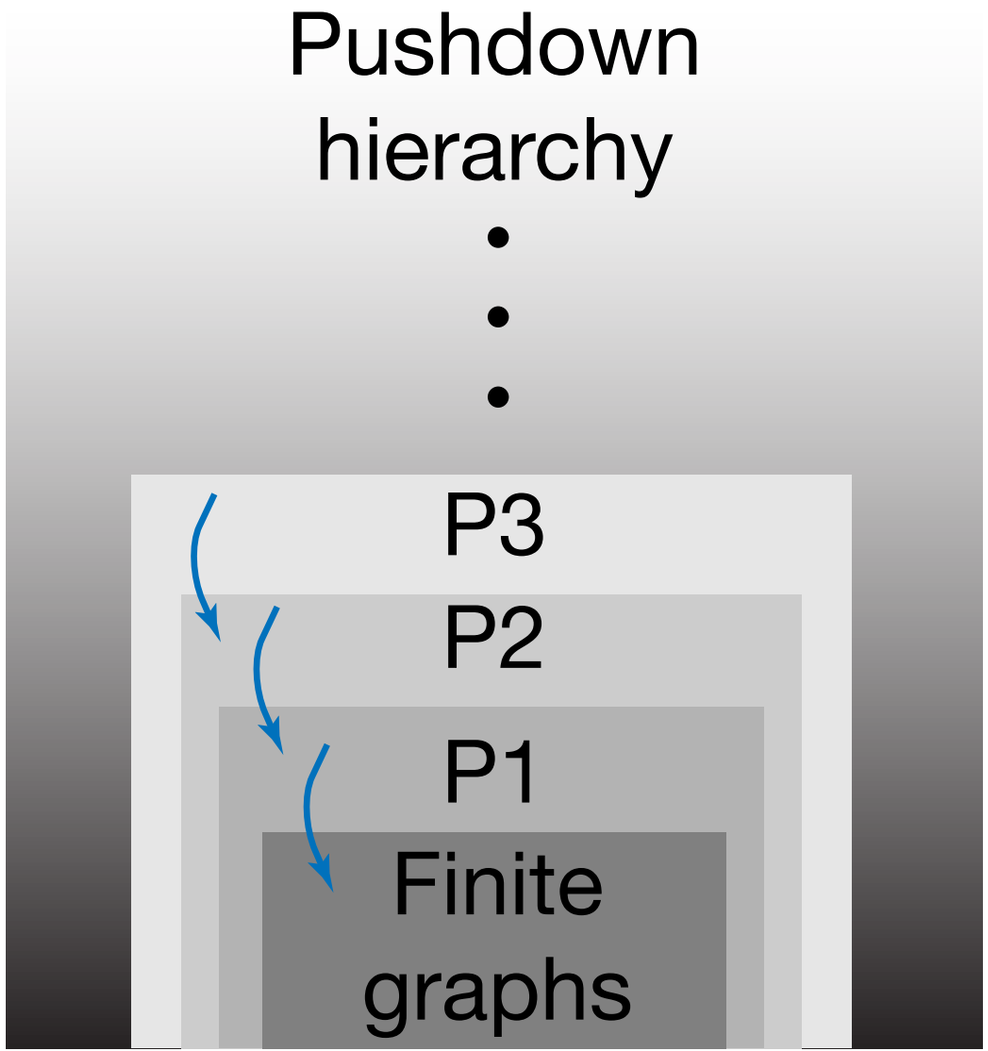
Pushdown hierarchy

Graphs obtained from finite graphs by iterating transductions and unfoldings.

Thm [Carayol, Wohrle '03]

$$\mathcal{T}(k-tree) \equiv \overbrace{\mathcal{T}(\,Unf(\ldots(\mathcal{T}(\,Unf}^{k}(finite\ \ graph)\ldots)$$

Remark: Every graph in the pushdown hierarchy has decidable MSOL theory



Higher-order pushdown automata [Maslov '74]

A **2-stack** is a stack of stacks. $[a_1^1 \dots a_{k_1}^1][a_1^2 \dots a_{k_2}^2] \dots [a_1^n \dots a_{k_n}^n]$

New operation of copying the top-most stack:

$$q[w_1] \dots [w_i] \to q[w_1][w_1] \dots [w_i].$$

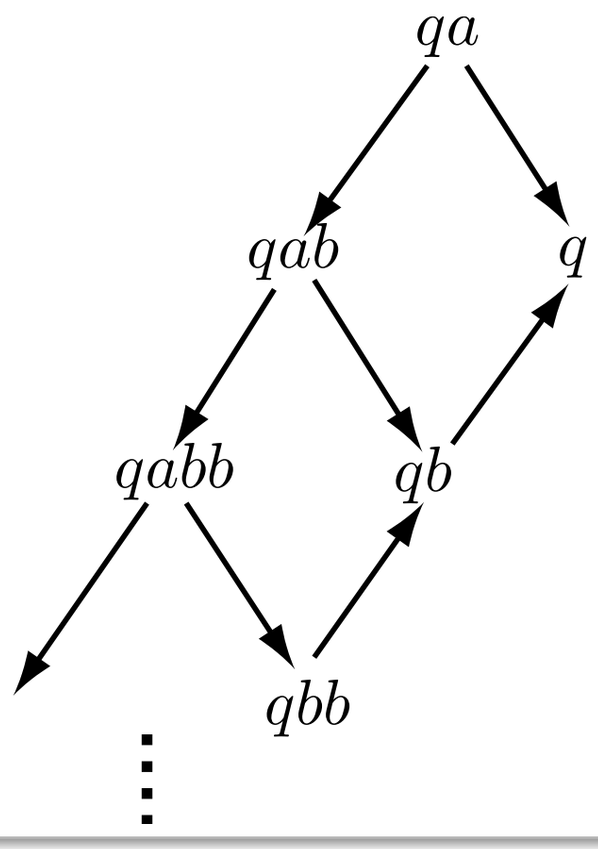
Remark: The 2-stack gives additional power.

One can, for example, recognise $a^n b^n c^n$.

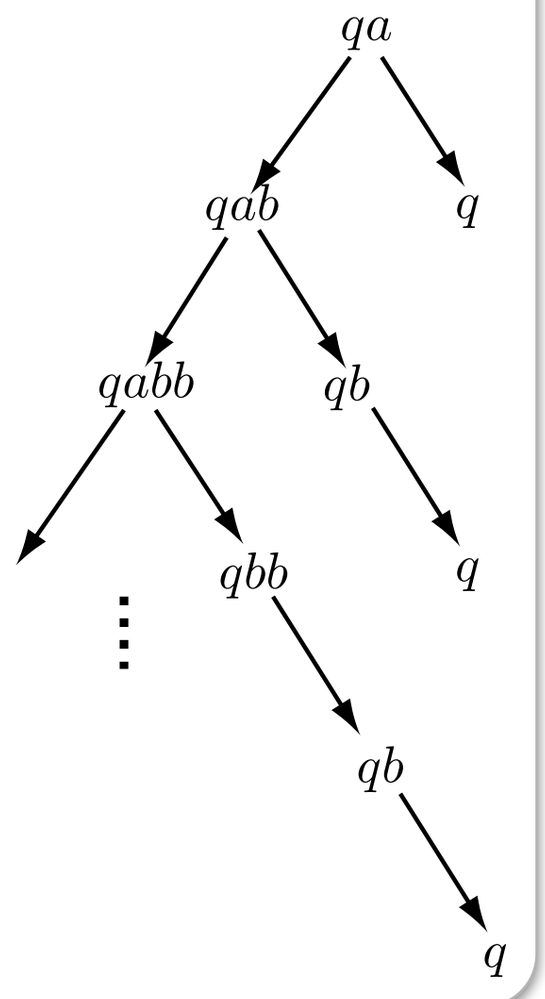
The **graph of configurations** of a machine:

- **nodes**: configurations of the machine
- **edges**: represent one step of the computation

Pushdown graph: the graph of configurations of a pushdown automaton.

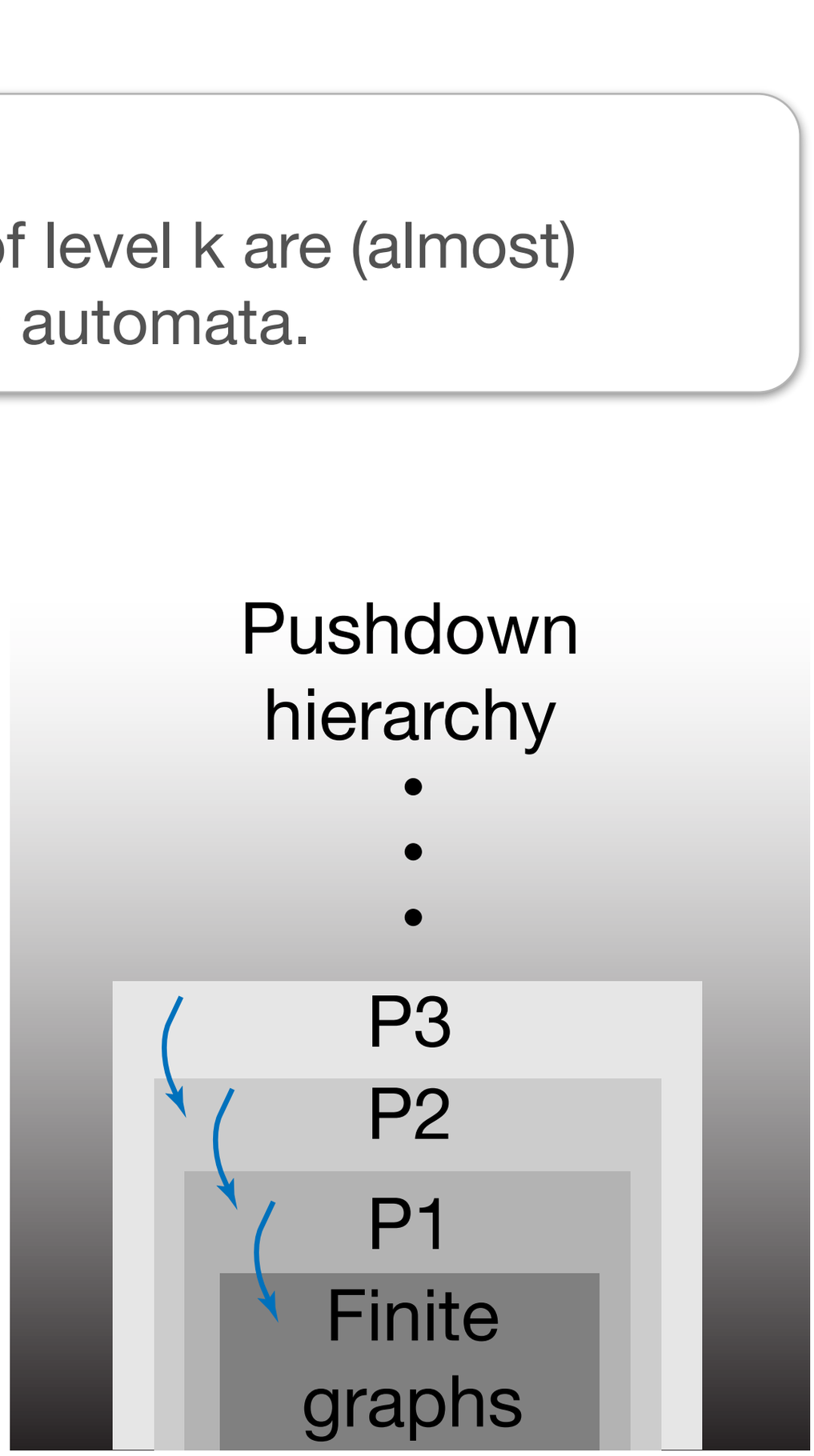


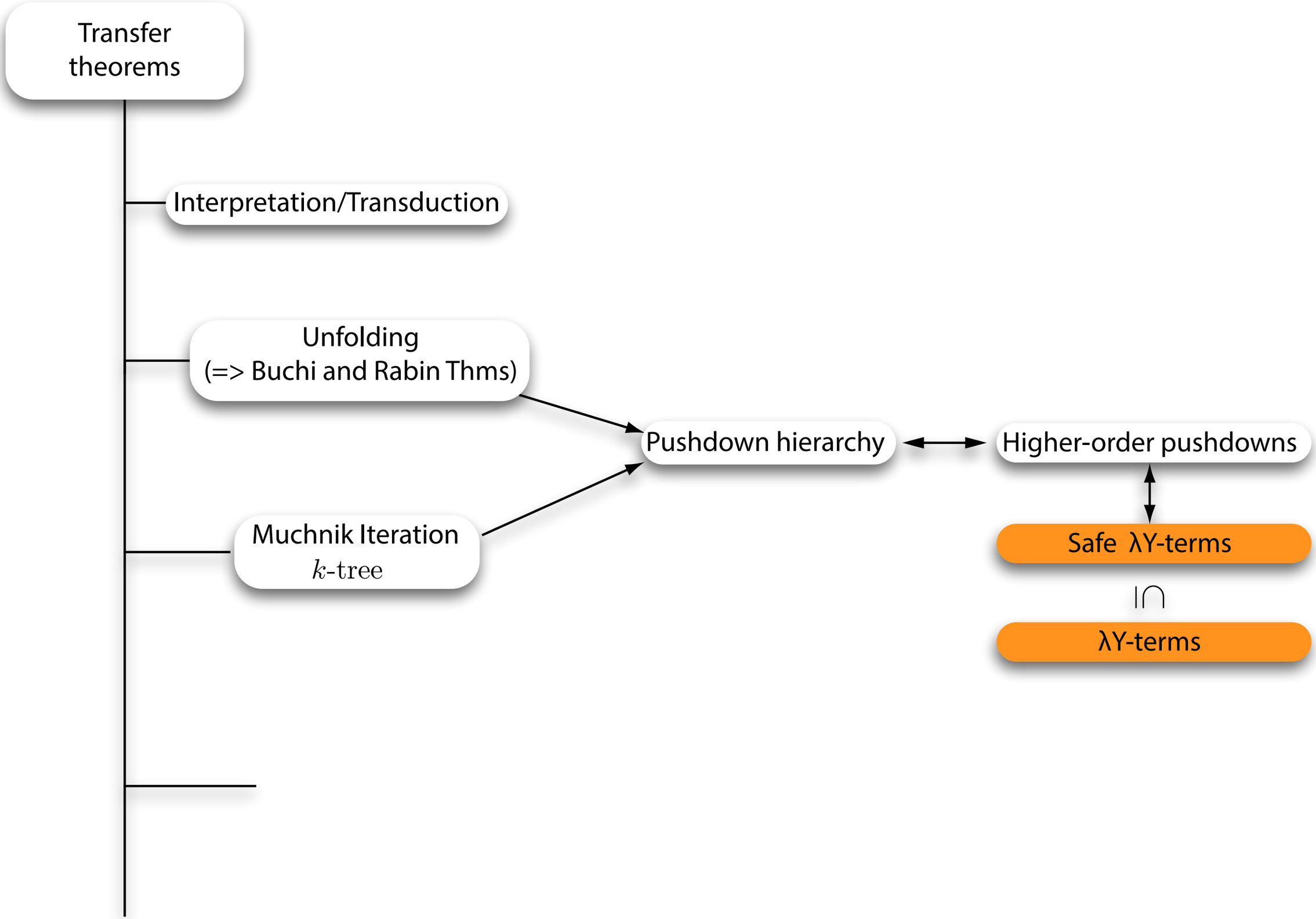
Pushdown tree: unfolding of a pushdown graph.



Thm [Carayol, Wohrle ‘03]
Graphs of the pushdown hierarchy of level k are (almost) configuration graphs of k-pushdown automata.

Thm [Engelfriet ‘91, Carayol & Wohrle ‘03]:
The pushdown hierarchy is strict.



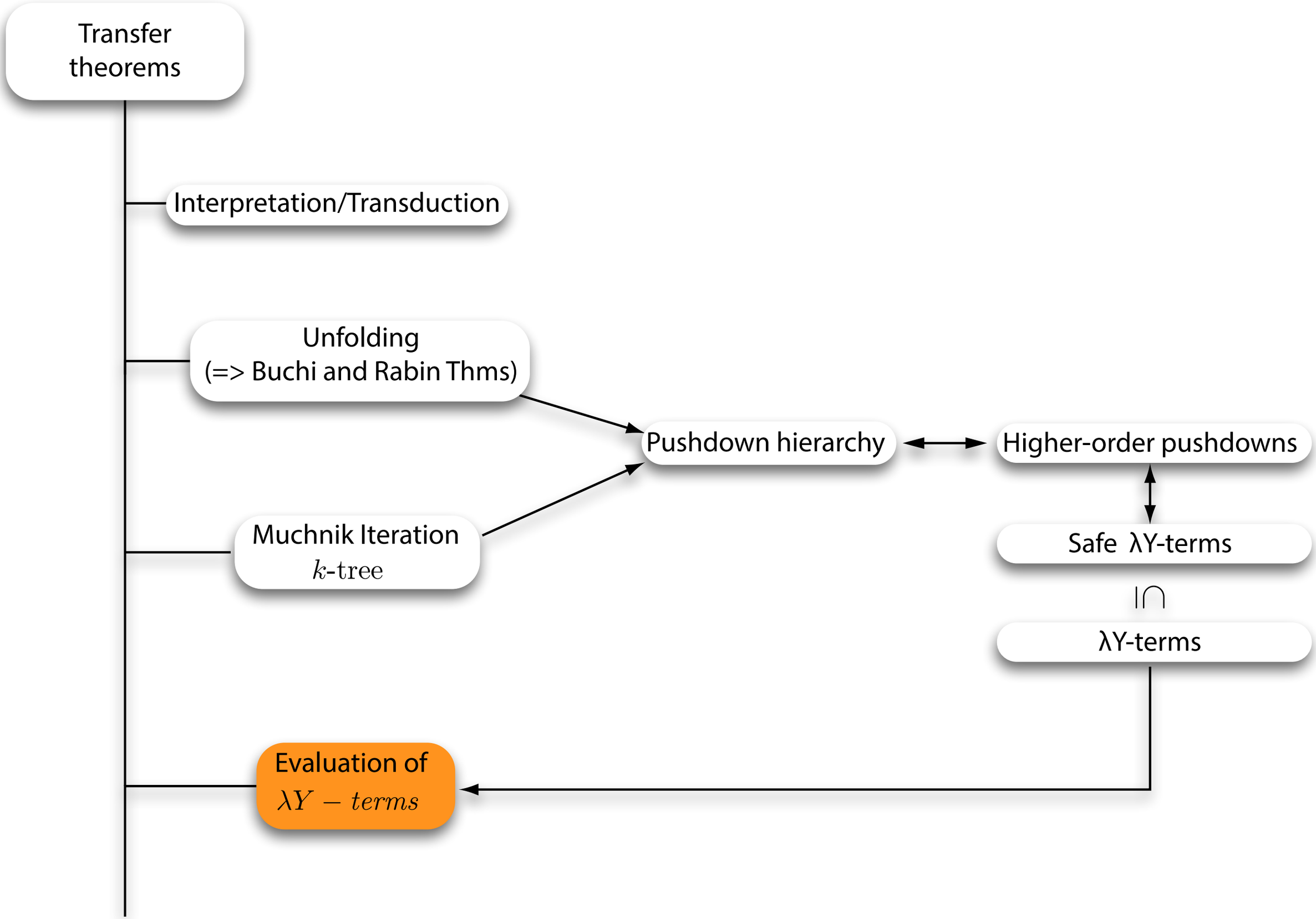

Transfer theorems
Interpretation/Transduction
Unfolding (=> Buchi and Rabin Thms)
Muchnik Iteration
k-tree
Pushdown hierarchy
Higher-order pushdowns
Safe λY-terms
∩
λY-terms

Order of a type: $Ord(\alpha \to \beta) = \max(Ord(\alpha) + 1, Ord(\beta))$

Thm [Courcelle '76]:
Böhm trees of 1-st order λY-terms ≈ pushdown trees.

Thm [Knapik, Niwinski, Urzyczyn '02]:
Böhm trees of n-th order ***safe*** λY-terms ≈ n-th order pushdown trees

Safe ≈ no parameters in recursion ≈ no problems with static links


Transfer theorems
Interpretation/Transduction
Unfolding (=> Buchi and Rabin Thms)
Muchnik Iteration k-tree
Pushdown hierarchy
Higher-order pushdowns
Safe λY-terms
∩
λY-terms
Evaluation of $\lambda Y - terms$

Thm [Ong ‘06]:
MSOL theory of the Bohm-tree of a λY-term is decidable.

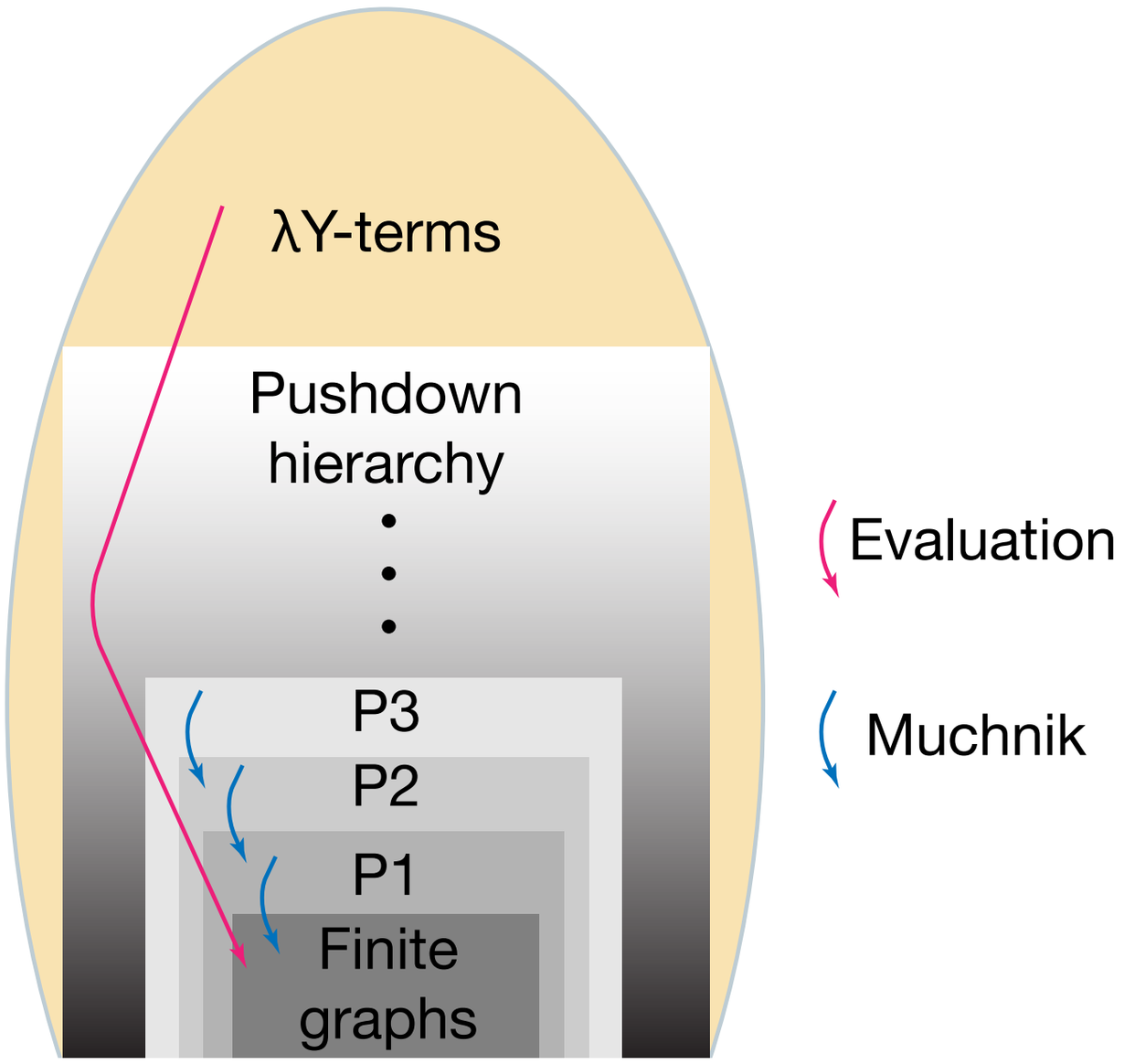
Thm [Parys ‘12]:
There is no safe λY-term having the same Bohm-tree as the Urzyczyn’s λY-term.

Thm [Salvati & W.’13]:
The operation of evaluation of λY-terms (over fixed finite signature) is MSO-compatible.

Transfer theorem:

Fix a finite signature (constants, variables, types).
For every MSO formula φ there is an MSOL formula $\widehat{\varphi}$ such that
for every λY-term M over the fixed signature:

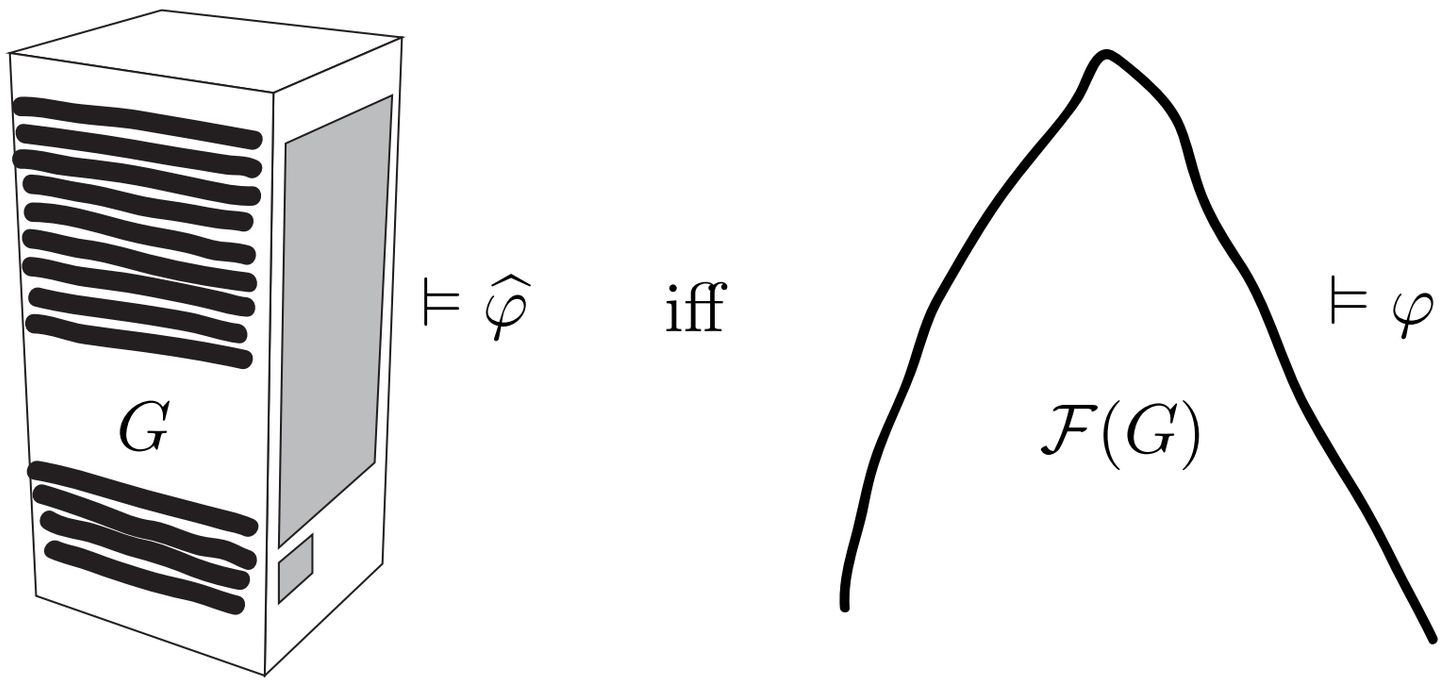
$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi.$$



Consider an operation $\mathcal{F}$ on graphs.

Transfer property for $\mathcal{F}$
For every φ one can effectively construct $\widehat{\varphi}$, s.t., for every G:

$$G \models \widehat{\varphi} \qquad \text{iff} \qquad \mathcal{F}(G) \models \varphi.$$



We say in this case that $\mathcal{F}$ is **MSO-compatible.**

Transfer theorem says that for fixed signature (constants, variables, types) the operation of evaluation is MSO-compatible.

Transfer theorems

Interpretation/Transduction

Unfolding
(=> Buchi and Rabin Thms)

Muchnik Iteration
k-tree

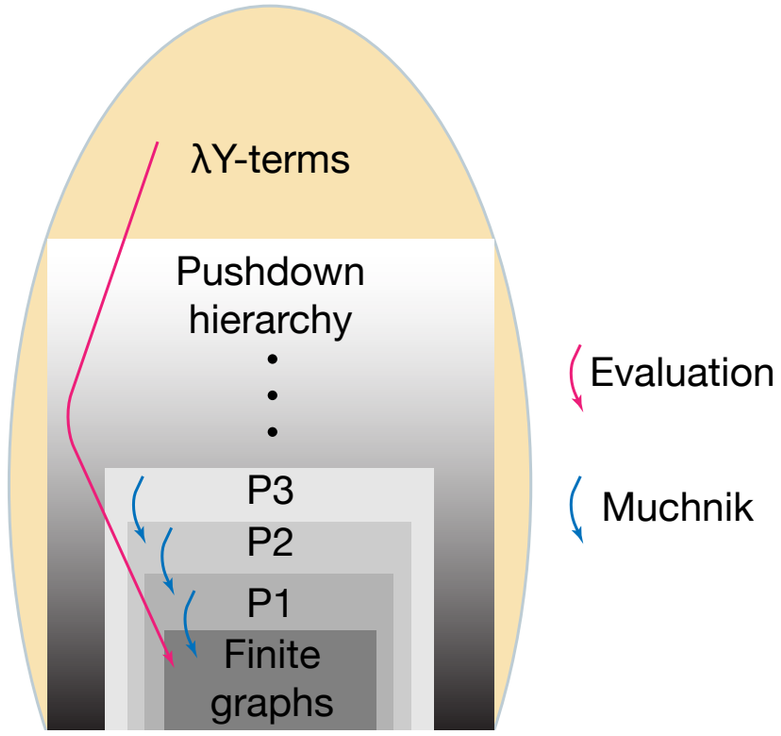
Pushdown hierarchy

Higher-order pushdowns

Safe λY-terms

∩

λY-terms

Evaluation of
$\lambda Y - terms$

Semantic evaluation