

Deterministic Linear Dynamic Logic and Hilbert's Choice Operator

Eugenia Ternovska

Simon Fraser University

Strategic Reasoning Workshop, September 7, 2020

Introduction: from Classical to Dynamic Logic over Paths

Goal: Formalize computation declaratively and understand **how** computation happens and **what** makes computations efficient

What can we take from classical logic for formalizing efficient computations, what do we need to drop, and what needs to be added?

In the process, we obtained a logic that, we think, is suitable for reasoning about **strategic behaviours in an environment**. We are going to present this logic here. But first, we explain how this logic was obtained.

The Starting Question

A typical question (model checking problem) for classical logics:

$$\mathfrak{A} \models \varphi?$$

What happens “inside” when formula φ and input structure $\mathfrak{A}$ are combined?

- ▶ Look into the internals of declarative computations
- ▶ Develop an “operational semantics” of classical logic

Analyze the ways information propagates

Find a way to **control the propagation**

Overview

- ▶ From FO(LFP), obtain a Dynamic Logic ($\approx$ mu-calculus $L\mu$)
- ▶ **Restrict** logical connectives ($\approx$ Description Logics)
- ▶ **Add** **Choice** = Strategy function ($\approx$ Hilbert's ε) and **Converse**

$$\mathbf{DL} \times \mathbf{DL} + \varepsilon + \smile$$

- ▶ Same Logic, Two Views:
 - ▶ Algebra of Binary Relations with Choice and Converse
 - ▶ 2WDetLDL_f : Two-Way Deterministic Linear Dynamic Logic over finite traces
- ▶ Equivalence of the two views
- ▶ Example
- ▶ Conclusion

“Operational Semantics” of Classical Logic

Start from FO(LFP), in an algebraic syntax ($\cup$, $\cap$, $\neg$, cyl, $\mu Z.\varphi$)
but with second-order variables

Partition variables of atomic predicate symbols into Inputs/Outputs

$$M(\bar{X}; \bar{Y}) \in \mathcal{M} \quad (\text{atomic module symbols})$$

atomic module (a transduction)

Also, **add composition**

An algebra of transductions

“Operational Semantics” of Classical Logic

Start from FO(LFP), in an algebraic syntax ($\cup, \cap, \neg, \text{cyl}, \mu Z.\varphi$)
but with second-order variables

Partition variables of atomic predicate symbols into Inputs/Outputs

$$M(\bar{X}; \bar{Y}) \in \mathcal{M} \quad (\text{atomic module symbols})$$

atomic module (a transduction)

Also, **add composition**

An algebra of transductions

This gives us the full **Logic of Information Flows (LIF)**

(will restrict and extend with ε and $\smile$)

Binary Semantics of LIF

Semantics:

an algebra of binary relations on τ -structures
over the same universe

(The domain of such an algebra consists of binary relations)

- ▶ De Morgan
- ▶ Peirce, then Schröder
- ▶ Tarski 1941
- ▶ Abstracted to relation algebra RA by Jónson and Tarski, then Givant
- ▶ Two recent big volumes by Givant
- ▶ More recently, Fletcher, Van den Bussche, Surinx et al.
- ▶ A paper with a historic overview by Pratt

Intuitively, **binary relations define a transition system**

Restricting the Operations of LIF: Determinization

Examine classical connectives:

- ▶ eliminate any non-determinism caused by the operations
e.g., cylindrifications ($\exists$) introduce “guess and check”,
iteration produces multiple outcomes, etc.
- ▶ ensure no prior computational work is wasted

e.g., intersection of two processes only “accepts” if the
outcomes agree, “rejects” otherwise, it required confluence

This ensures optimality of the set of connectives

Restricting the Operations: Determinization

Several operations, when applied to deterministic modules, introduce non-determinism.

- ▶ Complementation, α^-
- ▶ Unrestricted Union, $\alpha \cup \beta$
- ▶ Unrestricted Cylindrification, $\text{cyl}_\delta^{l,r}(\alpha)$
- ▶ Selection, $\sigma_{X=Y}^{l,r,lr}(\alpha)$
- ▶ Fixed point $\mu Z.\varphi$, even Iteration (Kleene star), α^*

Determinization

Several operations, when applied to deterministic modules, introduce non-determinism. **Restrict them:**

- ▶ Complementation, $\alpha^- \rightsquigarrow \{\curvearrowright \alpha, \curvearrowleft \alpha\}$ (like Intuitionistic $\neg$)
- ▶ Unrestricted Union, $\alpha \cup \beta \rightsquigarrow \alpha \sqcup \beta$
- ▶ Cylindrification, $\text{cyl}_\delta^{l,r}(\alpha) \rightsquigarrow \text{drop}$
- ▶ Selection, $\sigma_{X=Y}^{l,r,lr}(\alpha) \rightsquigarrow \sigma_{X=Y}^r(\alpha)$
- ▶ Iteration (Kleene star), $\alpha^* \rightsquigarrow \alpha^\uparrow$

Determinization

Several operations, when applied to deterministic modules, introduce non-determinism. **Restrict them:**

- ▶ Complementation, $\alpha^- \rightsquigarrow \{\curvearrowright \alpha, \curvearrowleft \alpha\}$ (like Intuitionistic $\neg$)
- ▶ Unrestricted Union, $\alpha \cup \beta \rightsquigarrow \alpha \sqcup \beta$
- ▶ Cylindrification, $\text{cyl}_\delta^{l,r}(\alpha) \rightsquigarrow \text{drop}$
- ▶ Selection, $\sigma_{X=Y}^{l,r,lr}(\alpha) \rightsquigarrow \sigma_{X=Y}^r(\alpha)$
- ▶ Iteration (Kleene star), $\alpha^* \rightsquigarrow \alpha^\uparrow$



$$\alpha ::= \text{id} \mid M(\bar{X}, \bar{Y}) \mid \overbrace{\curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha ; \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha)}^{\text{function-preserving}}$$

(Determinism-Preserving Fragment of the Logic of Information Flows)

Semantics of Deterministic Fragment (Informally)

$$\alpha ::= \text{id} \mid M(\underline{X}, \bar{Y}) \mid \overbrace{\neg\alpha \mid \neg\alpha \mid \alpha; \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha)}^{\text{function-preserving}}$$

Forward Unary Negation (Anti-Domain): $\neg\alpha$ – there is no outgoing α -transition ($\neg\alpha$ is similar)

Maximum Iterate: $\alpha^\uparrow$ – output the longest transition of α^*

Preferential Union: $\alpha \sqcup \beta$ – a determinization of $\cup$ – perform α if it is defined, otherwise perform β

Right Selection: $\sigma_{X=Y}^r(\alpha)$ – equality must hold in structures on the right side of the binary relation

Semantic Context T – Parameter of the Semantics

Atomic modules are **any binary relations on structures**

1. can be given **semantically** (e.g., via pattern matching in graph databases)
2. can be axiomatized **in some logic**:
 - ▶ specification of a (non-deterministic) action

In the second case, the **Law of Inertia** applies: what is not modified, remains the same

(imposed semantically)

A **semantic context** T gives: (1) a variable assignment to a vocabulary τ (an instantiation function), (2) a domain and (3) interpretation to module names M (without ε) in $\mathcal{M}$

Choice and Memory

In Determinism-Preserving Fragment,
the operations do not introduce any non-determinism

But atomic transitions can be non-deterministic

Something was still lacking in our formalization of computations:

Choice

- ▶ “choose the next element and continue”,
as in e.g. Gaussian Elimination algorithm
- ▶ “choose how to instantiate x in $\exists x\varphi(x)$ ”
in constructing a derivation in a proof system

The ability to remember what was done in the computation

Remaining Non-Determinism

In the Determinism-Preserving Fragment, given input structure $\mathfrak{A}$, Kripke models are tree-like because atomic modules are not required to be deterministic

$$\alpha ::= \text{id} \mid \underbrace{M(\bar{X}, \bar{Y})}_{\text{allow non-determ.}} \mid \overbrace{(\leadsto \alpha \mid \curvearrowright \alpha \mid \alpha ; \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha))}^{\text{function-preserving}}$$

Remaining Non-Determinism

In the Determinism-Preserving Fragment, given input structure $\mathfrak{A}$, Kripke models are tree-like because atomic modules are not required to be deterministic

$$\alpha ::= \text{id} \mid \underbrace{M(\bar{X}, \bar{Y})}_{\text{allow non-determ.}} \mid \overbrace{(\leadsto \alpha \mid \curvearrowright \alpha \mid \alpha ; \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha))}^{\text{function-preserving}}$$

But only one outcome is actually executed

The choice is due to the **environment** or due to the **program** (agent)

Add Hilbert's ε Operator and Converse

$$\alpha ::= \text{id} \mid (M(\underline{\bar{X}}, \bar{Y})) \mid \curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha ; \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha) \mid$$

Add Hilbert's ε Operator and Converse

add **dynamic choice** operator ε — **free** function variable
given history, decides on the outcome of atomic module

$$\alpha ::= \text{id} \mid \varepsilon(M(\bar{X}, \bar{Y})) \mid \curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha ; \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha) \mid$$

Add Hilbert's ε Operator and Converse

add **dynamic choice** operator ε — free function variable
given history, decides on the outcome of atomic module

$$\alpha ::= \text{id} \mid \varepsilon(M(\bar{X}, \bar{Y})) \mid \leadsto \alpha \mid \curvearrowright \alpha \mid \alpha ; \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha) \mid \alpha^\smile$$

add **converse**

Add Hilbert's ε Operator and Converse

add **dynamic choice** operator ε — free function variable
given history, decides on the outcome of atomic module

$$\alpha ::= \text{id} \mid \varepsilon(M(\bar{X}, \bar{Y})) \mid \leadsto \alpha \mid \curvearrowright \alpha \mid \alpha ; \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha) \mid \alpha^\smile$$

add **converse**

All information flows become **linear** and **logically reversible**

Add Hilbert's ε Operator and Converse

add **dynamic choice** operator ε — free function variable
given history, decides on the outcome of atomic module

$$\alpha ::= \text{id} \mid \varepsilon(M(\bar{X}, \bar{Y})) \mid \leadsto \alpha \mid \curvearrowright \alpha \mid \alpha ; \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha) \mid \alpha^\smile$$

add **converse**

All information flows become **linear** and **logically reversible**

We call it a Linear Logic of Information Flows (LinLIF)

Hilbert's Epsilon (ε)

Goal of Hilbert's program:

formalize mathematics and develop ways to mechanize proofs

Important idea: eliminate quantifiers by introducing notation

$\varepsilon_x A(x)$ for a witness of a quantifier [see Wirth: 2016]

$$\exists x A(x) \quad \Leftrightarrow \quad A(\varepsilon_x A(x))$$

Introduced by Hilbert and Bernays in 1939

Similarly to Hilbert, we specify a specific Choice (witness) abstractly

But we do it **dynamically** –

since, in any proof, we remember what was done in previous steps

Trees and Concrete Choice Function

Let $\mathbf{U}$ be the alphabet of all τ -structures over the same domain.

A **tree** over $\mathbf{U}$ is a (finite or infinite) nonempty set $Tr \subseteq \mathbf{U}^*$ such that for all $x \cdot c \in Tr$, with $x \in \mathbf{U}^*$ and $c \in \mathbf{U}$, we have $x \in Tr$.

(each node records the history)

Dynamic Choice function $f : Tr \rightarrow \mathbf{U}$ maps history to the next state

Given node x , f returns child $c \in \mathbf{U}$ such that $x \cdot c \in Tr$

For every Choice function f , there is a **unique branch** $b(f)$:

$$b(f) := \{\mathbf{e}, \mathbf{e} \cdot f(\mathbf{e}), \mathbf{e} \cdot f(\mathbf{e}) \cdot f(f(\mathbf{e})), \dots\}$$

Semantics of Choice and Converse

Recall: ε is a free function variable, f is a concrete choice function (a parameter of the semantics), T gives semantic context (atomic transitions)

Semantics of Epsilon

$$\llbracket \varepsilon M(\bar{X}; \bar{Y}) \rrbracket_T^f := \{ (w(last), f(w_{last})) \in \llbracket M(\bar{X}; \bar{Y}) \rrbracket_T \text{ for } w \text{ in } b(f) \}.$$

Semantics of Converse

$$\llbracket \varepsilon M(\bar{X}; \bar{Y})^\smile \rrbracket_T^f := \{ (\mathfrak{B}, \mathfrak{A}) \in \mathbf{U} \times \mathbf{U} \mid (\mathfrak{A}, \mathfrak{B}) \in \llbracket \varepsilon M(\bar{X}; \bar{Y}) \rrbracket_T^f \}.$$

Extend to all operators inductively:

$$\begin{aligned} (\alpha; \beta)^\smile &:= \beta^\smile; \alpha^\smile, & (\curvearrowright \alpha)^\smile &:= \curvearrowright \alpha, \\ (\curvearrowleft \alpha)^\smile &:= \curvearrowleft \alpha, & (\alpha^\uparrow)^\smile &:= \alpha^\smile{}^\uparrow, \\ (\alpha^\smile)^\smile &:= \alpha, & (\sigma_\Theta^r(\alpha))^\smile &:= \sigma_\Theta^r(\alpha^\smile). \end{aligned}$$

Satisfaction Relation

Binary Satisfaction Relation

$(\mathfrak{A}, \mathfrak{B})$ **satisfies** α under variable assignment s and choice function f :

$$(\mathfrak{A}, \mathfrak{B}) \models_T \alpha[\varepsilon/f] \Leftrightarrow (\mathfrak{A}, \mathfrak{B}) \in \llbracket \alpha \rrbracket_T^f$$

Unary Satisfaction Relation

$$\mathfrak{A} \models_T \text{Dom}(\alpha)[\varepsilon/f] \Leftrightarrow (\mathfrak{A}, \mathfrak{A}) \in \llbracket \text{Dom}(\alpha) \rrbracket_T^f$$

where $\text{Dom}(\alpha) := \curvearrowright \curvearrowright (\alpha)$

2-Way Deterministic Linear Dynamic Logic (DetLDL_f)

A two-sorted syntax of LinLIF:

$$\begin{aligned}\alpha &::= \text{id} \mid \varepsilon M_a \mid \alpha ; \alpha \mid \curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha) \mid \alpha^\sim \mid \varphi? \\ \varphi &::= \mathbf{T} \mid M_p \mid \neg \varphi \mid |\alpha\rangle \varphi \mid \langle \alpha | \varphi\end{aligned}$$

where α is a process formula, φ is a state formula

The second line are shorthands:

$$\begin{aligned}\mathbf{T} &:= \text{id} \\ |\alpha\rangle \varphi &:= \curvearrowright \curvearrowright (\alpha ; \varphi) && (= \text{Dom}(\alpha ; \varphi)) \\ \langle \alpha | \varphi &:= \curvearrowleft \curvearrowleft (\varphi ; \alpha) && (= \text{Img}(\alpha ; \varphi)) \\ \neg \varphi &:= \curvearrowright \varphi \text{ (or } \curvearrowleft \varphi) && (\text{standard modal negation}) \\ \varphi? &:= \curvearrowright \curvearrowright \varphi && (\text{test action})\end{aligned}$$

Some Properties

Properties of the Intuitionistic negation: $\neg\neg\neg\neg\alpha = \neg\neg\alpha$
 $\neg\neg\neg\alpha \neq \alpha$

We also define:

$$[\alpha] \varphi := \neg[\alpha] \neg\varphi \quad (= \neg(\alpha; \neg\varphi))$$

and $\text{Dom}(\alpha) := \neg\neg\neg\alpha$.

Inside the two modalities, can have imperative programs (definable)

Definability of Programming Constructs

$\sqcup$ and $\uparrow$ are restrictions of $\cup$ and $*$ $\Rightarrow$ **sub-regular languages**

Nevertheless, the main programming constructs are still definable:

```
skip := id,  
fail :=  $\neg$ id,  
if  $\varphi$  then  $\alpha$  else  $\beta$  :=  $(\varphi? ; \alpha) \sqcup \beta$ ,  
while  $\varphi$  do  $\alpha$  :=  $(\varphi? ; \alpha)^\uparrow ; (\neg\varphi?)$ ,  
repeat  $\alpha$  until  $\varphi$  :=  $\alpha ; ((\neg\varphi?) ; \alpha)^\uparrow ; \varphi?$ .
```

In addition, can look “backwards in time” (we’ll see how it is used)

$\Rightarrow$ **2-way Deterministic Regular Languages**

Toward Path Semantics

A two-sorted syntax of LinLIF:

$$\begin{aligned}\alpha &::= \text{id} \mid \varepsilon M_a \mid \alpha ; \alpha \mid \curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha \sqcup \alpha \mid \alpha^\uparrow \mid \sigma_{X=Y}^r(\alpha) \mid \alpha^\sim \mid \varphi? \\ \varphi &::= \mathbf{T} \mid M_p \mid \neg \varphi \mid |\alpha\rangle \varphi \mid \langle \alpha | \varphi\end{aligned}$$

A **path** is a finite word $\mathbf{p} \subseteq \mathbf{U}^*$

$$\mathfrak{A} = \mathfrak{A}_0 \cdot \mathfrak{A}_2 \cdot \mathfrak{A}_3 \cdot \dots \cdot \mathfrak{A}_m = \text{last}(\mathbf{p})$$

Path semantics defines

- ▶ “true between states $\mathfrak{A}$ and $\mathfrak{B}$ on path $\mathbf{p}$ ”
— relation $\mathbf{p}(i, j) \in \mathcal{L}(\alpha)$
- ▶ “true in state $\mathfrak{A}$ on path $\mathbf{p}$ ”

Connection between Path and Binary Semantics

A path $\mathbf{p}$ is an **f -path** if $\mathbf{p} \in b(f)$, the unique branch in Tr on $\mathbf{U}$ associated with a concrete Choice function f

Lemma

For any process expression α , we have:

$$\mathbf{p}(i, j) \in \mathcal{L}(\alpha) \iff (\mathbf{p}(i), \mathbf{p}(j)) \in \llbracket \alpha \rrbracket_T^f,$$

where $\mathbf{p}$ is an f -path for some concrete f

Theorem

For any state formula φ , we have:

$$\mathbf{p}, 0 \models_T \varphi \iff \mathfrak{A} \models_T \varphi[\varepsilon/f],$$

where $\mathbf{p}$ is an f -path for some concrete f and $\mathbf{p}(0) = \mathfrak{A}$

Formalization of a Decision Problem

Recall that semantic context T provides variable assignment $s : \mathbb{V} \rightarrow \tau$, a domain and an interpretation of atomic symbols

Define the last element in a path: $\text{Last} := [id] \neg \mathbf{T}$

Problem: **Main Task (Decision Version)**

Given: τ -structure $\mathfrak{A}$, e.g., a graph, expression α and semantic context T .

Decide: $\mathfrak{A} \models_T [\alpha] \text{ Last}$? Recall: α may contain ε

Witnesses: finite sequences of τ -structures over the input domain

$$\mathfrak{A} = \mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \dots, \mathfrak{A}_m = \mathfrak{A}_{\text{last}(\mathbf{p})}$$

Each sequence corresponds to a particular instantiation f of the free choice function variable ε

$\Rightarrow$ **Formal Language Non-Emptiness problem:** $\mathcal{L}(\alpha) \neq \emptyset$

Complexity of the Main Task

Theorem: For each specific choice function, the data complexity of

$$\mathfrak{A} \models_T |\alpha\rangle\text{Last?}$$

for LinLIF [M-CQ] is **in PTIME**

i.e., LinLIF where atomic modules are

Output-Monadic Conjunctive Queries [M-CQ]

(data complexity means that the formula is fixed)

Proof: Via Structural Operational Semantics [Plotkin:1981]

An Abbreviation

To reason backwards in time, define:

Back-Exists: $\mathbf{BE}\varphi := \langle (\text{repeat } any_M \text{ until } \varphi)^\smile \mid \mathbf{T} \rangle$

Back-Globally: $\mathbf{BG}\varphi := \smile \mathbf{BE} \smile \varphi$

(similar to P and H , respectively, of Arthur Prior's Tense Logic and a version of LTL with backwards-facing temporal modalities)

Problem: **EVEN** α_E

Given: A set represented by a structure $\mathfrak{A}$ with an empty vocab.
Question: Is $|dom(\mathfrak{A})|$ even?

(not in MSO, not in Datalog, not in FO(LFP) unless an order is given)

$\alpha_E := \mathbf{repeat} \ (GuessNewO ; GuessNewE) \ \mathbf{until} \ \neg GuessNewO$

$$\begin{aligned} GuessP &:= \varepsilon \ \{ \ P(x) \leftarrow \quad \} \\ CopyPO &:= \{ \ O(x) \leftarrow \underline{P(x)} \} \\ CopyPE &:= \{ \ E(x) \leftarrow \underline{P(x)} \} \end{aligned}$$
$$\begin{aligned} GuessNewO &:= (GuessP ; \mathbf{BG}(\sigma_{P \neq E}^r(\sigma_{P \neq O}^r(\text{id})))) ; CopyPO \\ GuessNewE &:= (GuessP ; \mathbf{BG}(\sigma_{P \neq E}^r(\sigma_{P \neq O}^r(\text{id})))) ; CopyPE \end{aligned}$$

$\mathfrak{A} \models |\alpha_E\rangle\text{Last}$ iff the size of the input domain is even

Conclusion

- ▶ Defined a logic that is, simultaneously:
 - ▶ an algebra of binary relations with Choice, and
 - ▶ a dynamic logic under a path semantics
- ▶ Agent's programs are deterministic, Choice function determines the outcomes of atomic actions
- ▶ Show: Imperative programming constructs are definable even though the non-determinism of regular expressions is avoided
- ▶ More expressive than MSO on unordered structures
- ▶ Future research: automata for DetLDL_f .

Thank you!

Extra Slides: Binary Semantics of Algebraic Operations

(subscript f can be ignored for now)

Identity (Diagonal) $\llbracket \text{id} \rrbracket_T^f := \{(\mathfrak{A}, \mathfrak{B}) \in \mathbf{U} \times \mathbf{U} \mid \mathfrak{A} = \mathfrak{B}\}.$

Right Selection (Equality)

$$\llbracket \sigma_{X=Y}^r(\alpha) \rrbracket_T^f := \{(\mathfrak{A}, \mathfrak{B}) \in \mathbf{U} \times \mathbf{U} \mid (\mathfrak{A}, \mathfrak{B}) \in \llbracket \alpha \rrbracket_T^f \\ \text{and } (s(X))^{\mathfrak{B}} = (s(Y))^{\mathfrak{B}}\}.$$

Sequential Composition

$$\llbracket \alpha ; \beta \rrbracket_T^f := \{(\mathfrak{A}, \mathfrak{B}) \in \mathbf{U} \times \mathbf{U} \mid \exists \mathfrak{C} ((\mathfrak{A}, \mathfrak{C}) \in \llbracket \alpha \rrbracket_T^f \text{ and } (\mathfrak{C}, \mathfrak{B}) \in \llbracket \beta \rrbracket_T^f)\}.$$

Forward Unary Negation (Anti-Domain)

$$\llbracket \neg \alpha \rrbracket_T^f = \{(\mathfrak{B}, \mathfrak{B}) \in \mathbf{U} \times \mathbf{U} \mid \forall \mathfrak{B}' (\mathfrak{B}, \mathfrak{B}') \notin \llbracket \alpha \rrbracket_T^f\}.$$

Backwards Unary Negation (Anti-Image)

$$\llbracket \neg \alpha \rrbracket_T^f = \{(\mathfrak{B}, \mathfrak{B}) \in \mathbf{U} \times \mathbf{U} \mid \forall \mathfrak{B}' (\mathfrak{B}', \mathfrak{B}) \notin \llbracket \alpha \rrbracket_T^f\}.$$

Maximum Iterate

$$\alpha^\uparrow = \bigcup_{1 \leq n < \omega} \alpha_n,$$

where $\alpha_1 := \alpha; \curvearrowright \alpha$, $\alpha_{n+1} := \alpha; \alpha_n$.

(an alternative definition: $\alpha^\uparrow := \mu Z.(\curvearrowright \alpha \cup \alpha; Z).$)

Preferential union $\llbracket \alpha \sqcup \beta \rrbracket_T^f = \left\{ (\mathfrak{A}, \mathfrak{B}) \mid \right.$

$$\left. \begin{array}{ll} (\mathfrak{A}, \mathfrak{B}) \in \llbracket \alpha \rrbracket_T^f & \text{if } (\mathfrak{A}, \mathfrak{A}) \in \llbracket \text{Dom}(\alpha) \rrbracket_T^f, \\ (\mathfrak{A}, \mathfrak{B}) \in \llbracket \beta \rrbracket_T^f & \text{if } (\mathfrak{A}, \mathfrak{A}) \notin \llbracket \text{Dom}(\alpha) \rrbracket_T^f \\ & \text{and } (\mathfrak{A}, \mathfrak{A}) \in \llbracket \text{Dom}(\beta) \rrbracket_T^f \end{array} \right\},$$

where $\text{Dom}(\alpha) := \curvearrowright \curvearrowright \alpha$

Extra Slides: Connections to Other Logics

On finite strings:

LDL_f
regular
MSO on fin. strings

LTL_f
*-free regular
FO on fin. strings

“forward” LinLIF[M-UCQ]
det.regular
? on fin. strings

imperative constructs

Extra Slides: Paths

Let a **path** be a finite word $\mathbf{p} \subseteq \mathbf{U}^*$. Each path has the following form, where $\mathfrak{A}$ is the input structure.

$$\mathfrak{A} = \mathfrak{A}_0 \cdot \mathfrak{A}_2 \cdot \mathfrak{A}_3, \dots \cdot \mathfrak{A}_m = \text{last}(\mathbf{p}). \quad (1)$$

We call a path $\mathbf{p}$ an **f -path** if $\mathbf{p} \in b(f)$, the unique branch associated with the concrete Choice function f .

The **positions** in $\mathbf{p}$ are $i, j \in \mathbb{N}$, for $0 \leq i, j \leq \text{last}(\mathbf{p})$

and **structures** in those positions are $\mathbf{p}(i)$

$\mathbf{p}(i, j)$ denotes the **subword** of $\mathbf{p}$ starting from position i and terminating in position j , where $0 \leq i \leq j \leq \text{last}(\mathbf{p})$.