Project SPACMODL

Semantic Stream Processing in Business Auditing

Stephan Scheele

Informatics Theory Group
University of Bamberg

Joint work with Michael Mendler.
SYNCHRON 2008

1st - 5th December, 2008

Introduction & context

Research Project
“SPACMoDL”

- Funded by the German Research Council (DFG)
- Started June 2008
- Topic: Investigate logics and semantic programming models for auditing
- Collaboration with industrial partners

Area of interest: Auditing

- Verification of business transactions & data
- Mass data processing needs efficient stream processing procedures
- Audit problems:
  - Purchasing: Analysis of behaviour of purchasers, completed and future orders, ...
  - Accounts Payable: Open-item accounting, supplier ranking, ...
  - ...

- Offline Auditing: Auditing on database extracts (file streams)
- Online Auditing: Auditing in-place on information streams
  - act as intelligent audit procedures within an information stream
  - process transactions in real-time (as they come in)

Area of interest: Auditing

- Verification of business transactions & data
- Mass data processing needs efficient stream processing procedures
- Audit problems:
  - Purchasing: Analysis of behaviour of purchasers, completed and future orders, ...
  - Accounts Payable: Open-item accounting, supplier ranking, ...
  - ...

- Offline Auditing: Auditing on database extracts (file streams)
- Online Auditing: Auditing in-place on information streams
  - act as intelligent audit procedures within an information stream
  - process transactions in real-time (as they come in)

Classical Approach

- Spreadsheet-based, does not scale to large volume of data
- Database-based, specific and isolated applications
- Domain specific languages, ACL and Idea
  - Old-fashioned languages
  - Process mass-data stream based
  - Not strongly typed, not type-safe, only flat types
  - Modern features missing: Modularity, components, static typing
  - Do not utilize modern hardware and parallelism (Multicore CPUs)

Project objectives

- DSL for Auditing: Functional, declarative stream-processing language like Lustre
- Utilize Description Logics: Semantic interpretation of information streams
- Integrate Description Logic-reasoning services for advanced typing and knowledge-based data analysis
- Bring techniques from synchronous languages into the auditing world: component orientation, correct by construction, static typing, clear semantics, formal verification, clocktypes for optimization . . .
- Auditing is not (in every case) real-time critical but business critical

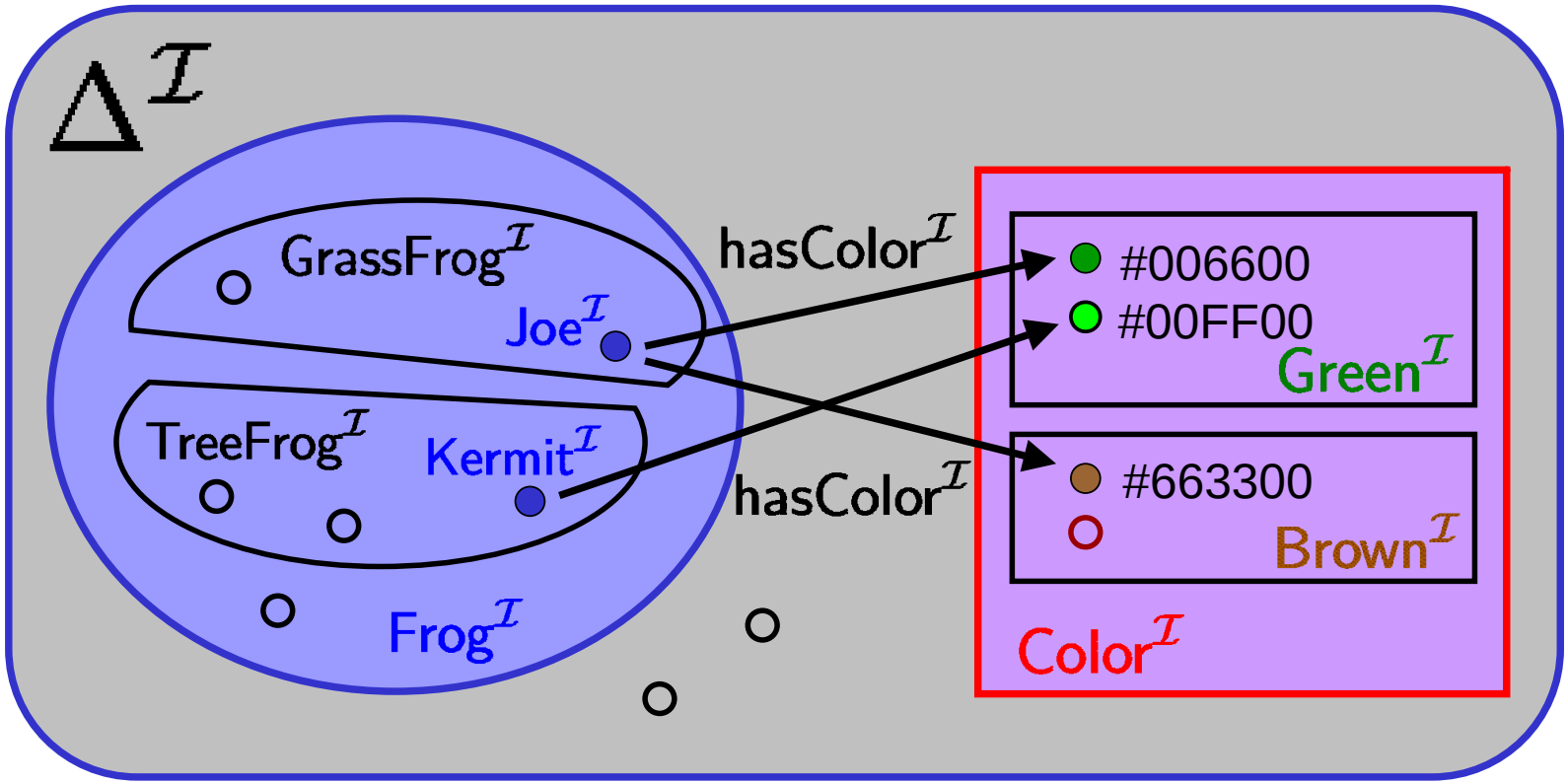
Description Logics

Family of logic based formalisms for the purpose of knowledge representation
Well-suited for the representation of and reasoning about

- terminological knowledge
- ontologies
- database schemata
- ...

- related to modal logic
- guarded fragment of predicate logic
- model theoretic semantics
- decidable decision problem

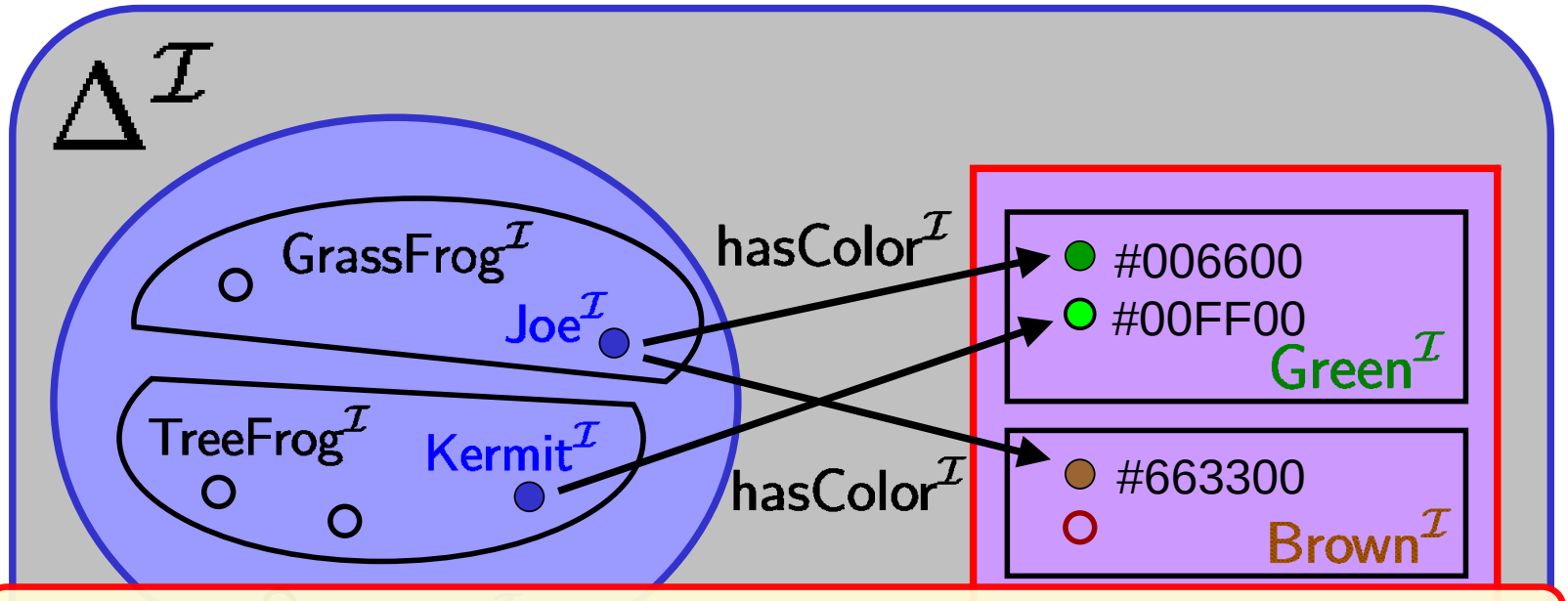
Example: Syntax & Semantics of $\mathcal{ALC}$



Atomic concepts: Frog, GrassFrog, TreeFrog, Colour, Green, Brown

Roles: hasColour

Example: Syntax & Semantics of $\mathcal{ALC}$



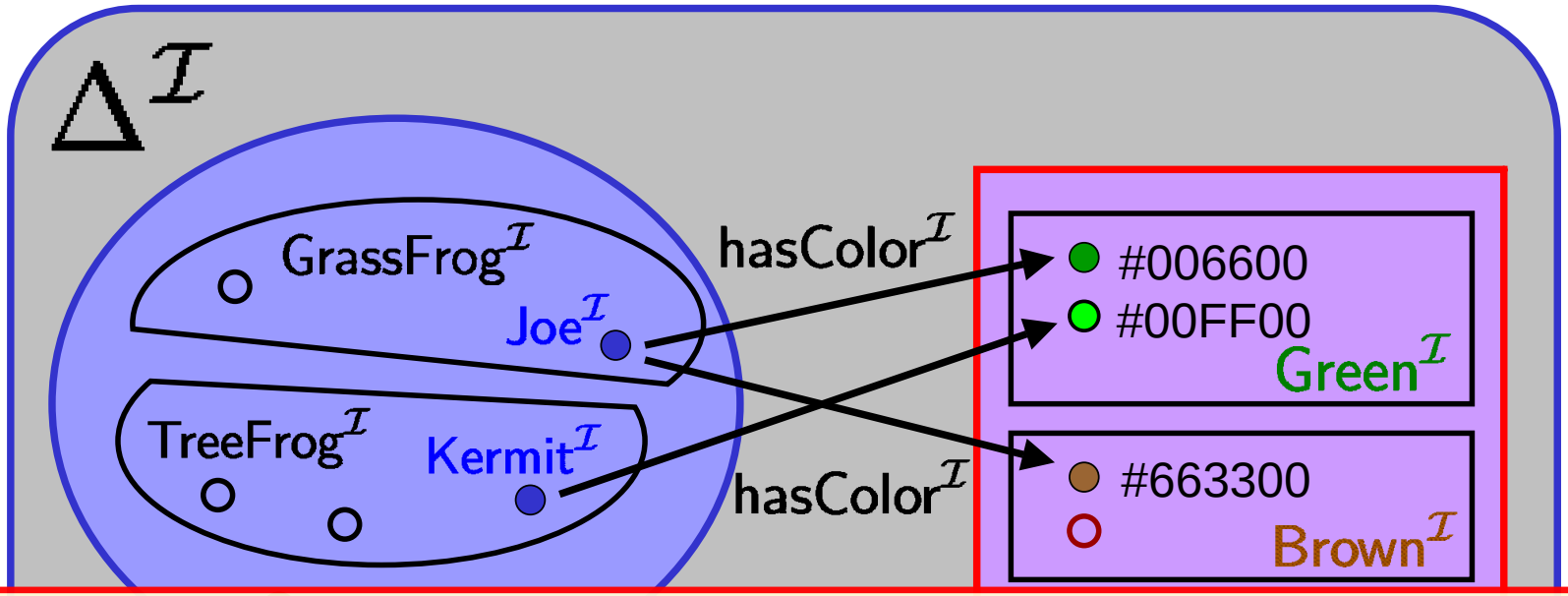
TBox statements:

$Frog \sqsubseteq Animal$

$GrassFrog \sqsubseteq Frog \sqcap \exists hasColor.Brown$

$TreeFrog \sqsubseteq Frog \sqcap \exists hasColor.Green$

Example: Syntax & Semantics of $\mathcal{ALC}$



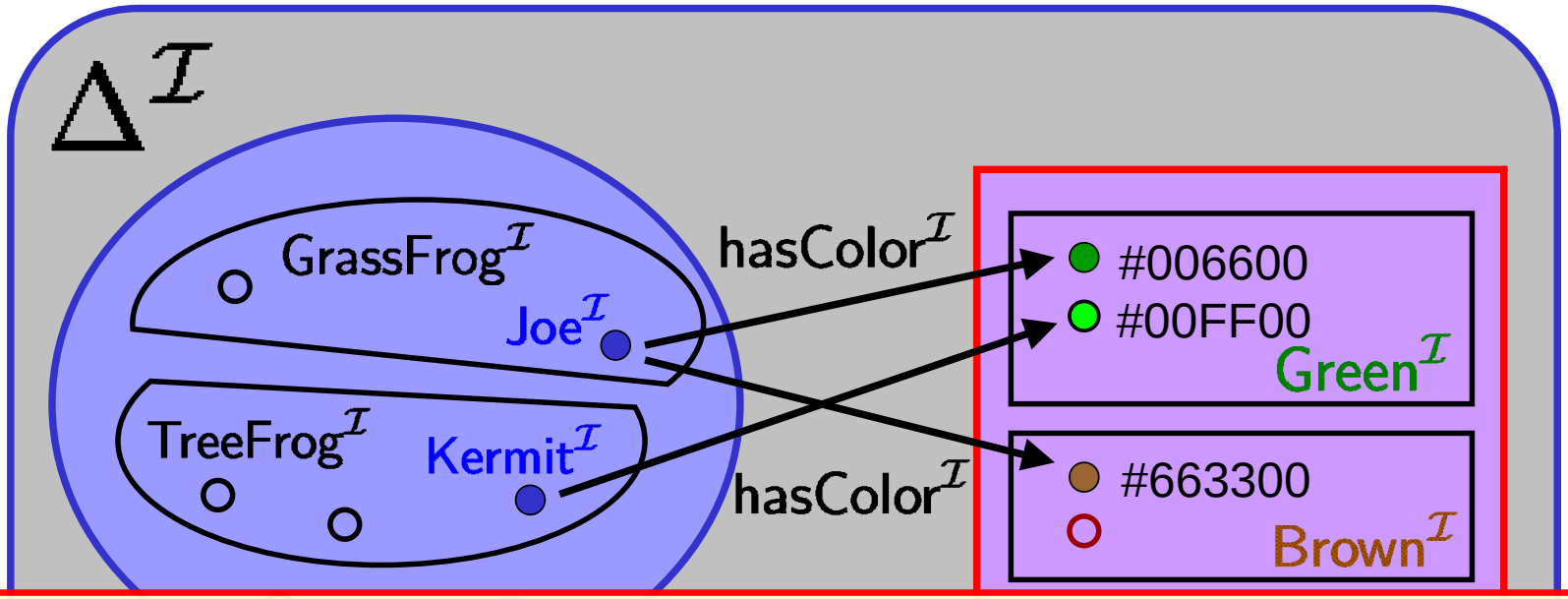
DL uses a variable free syntax:

$GrassFrog \sqsubseteq Frog \sqcap \exists hasColor.Brown$

can be translated into:

$\forall x.GrassFrog(x) \Rightarrow Frog(x) \wedge \exists y.hasColor(x,y) \wedge Brown(y)$

Example: Syntax & Semantics of $\mathcal{ALC}$



ABox statements:

Kermit:*TreeFrog*, *Joe*:*GrassFrog*

Kermit:$\forall$*hasColor*.*Green*

Joe, *Kermit*:$\exists$*hasColor*.*Green* $\sqcup$ $\exists$*hasColor*.*Brown*

Syntax and semantics of $\mathcal{ALC}$

Elementary descriptions:

- atomic concepts
- atomic roles

Concepts and roles are given standard Tarski-style model-theoretic semantics, their meaning is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with

- $\Delta^{\mathcal{I}}$ as the universe of individuals and
- an interpretation function $\cdot^{\mathcal{I}}$ mapping

Syntax and semantics of $\mathcal{ALC}$

Elementary descriptions:

- atomic concepts
- atomic roles

Concepts and roles are given standard Tarski-style model-theoretic semantics, their meaning is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with

- $\Delta^{\mathcal{I}}$ as the universe of individuals and
- an interpretation function $\cdot^{\mathcal{I}}$ mapping

Syntax and semantics of $\mathcal{ALC}$

Elementary descriptions:

- atomic concepts
- atomic roles

Concepts and roles are given standard Tarski-style model-theoretic semantics, their meaning is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with

- $\Delta^{\mathcal{I}}$ as the universe of individuals and
- an interpretation function $\cdot^{\mathcal{I}}$ mapping

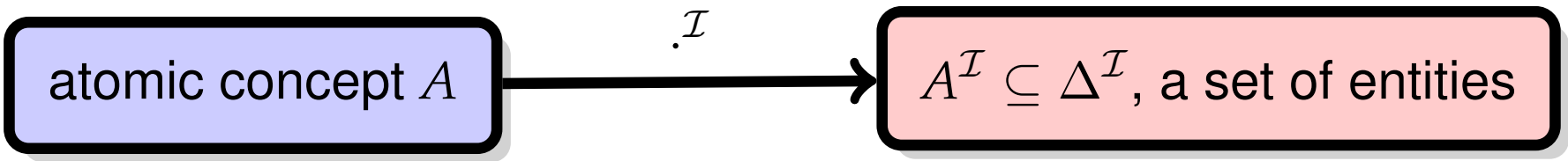
Syntax and semantics of $\mathcal{ALC}$

Elementary descriptions:

- atomic concepts
- atomic roles

Concepts and roles are given standard Tarski-style model-theoretic semantics, their meaning is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with

- $\Delta^{\mathcal{I}}$ as the universe of individuals and
- an interpretation function $\cdot^{\mathcal{I}}$ mapping



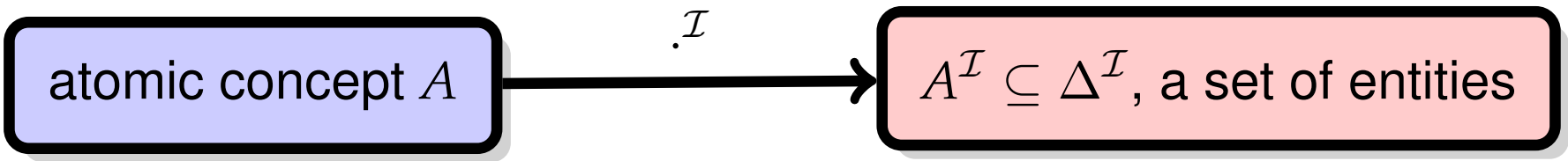
Syntax and semantics of $\mathcal{ALC}$

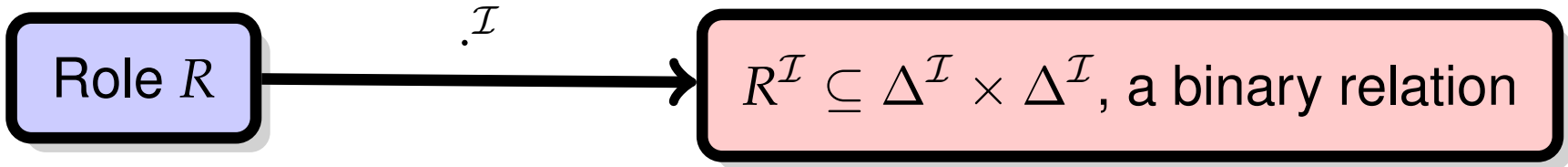
Elementary descriptions:

- atomic concepts
- atomic roles

Concepts and roles are given standard Tarski-style model-theoretic semantics, their meaning is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with

- $\Delta^{\mathcal{I}}$ as the universe of individuals and
- an interpretation function $\cdot^{\mathcal{I}}$ mapping





Description Logic specifications as stream types

- Business data come typically as streams of information, e.g. linearised database tables (streaming records)
- Considering streams as abstract entities
  - DL concepts can act as typing system and
  - specify semantical properties of stream elements

Typing streams (I)

Consider $\Delta^{\mathcal{I}} = \mathbb{D}^* \cup \mathbb{D}^\infty$ with $\mathbb{D} = \mathbb{N} \uplus \mathbb{B} \uplus (\mathbb{N} \times \mathbb{B})$ the universe of booleans, naturals and their pairings.

Typing streams (I)

Consider $\Delta^{\mathcal{I}} = \mathbb{D}^* \cup \mathbb{D}^\infty$ with $\mathbb{D} = \mathbb{N} \uplus \mathbb{B} \uplus (\mathbb{N} \times \mathbb{B})$ the universe of booleans, naturals and their pairings.
Refinement $\preceq^{\mathcal{I}}$ (time shift) is the (inverse) suffix ordering

$$\frac{v \in \mathbb{D}}{v \cdot s \preceq^{\mathcal{P}} s}$$

where $v \cdot s$ is the stream $s \in \mathbb{D}^\omega$ prefixed by value $v \in \mathbb{D}$.
For instance,

$$1 \cdot (2, \mathtt{T}) \cdot \mathtt{T} \cdot \mathtt{F} \preceq^{\mathcal{I}} (2, \mathtt{T}) \cdot \mathtt{T} \cdot \mathtt{F} \preceq^{\mathcal{I}} \mathtt{T} \cdot \mathtt{F} \preceq^{\mathcal{I}} \mathtt{F} \preceq^{\mathcal{I}} \epsilon,$$

where ϵ is the empty stream.

Typing streams (I)

Consider $\Delta^{\mathcal{I}} = \mathbb{D}^* \cup \mathbb{D}^\infty$ with $\mathbb{D} = \mathbb{N} \uplus \mathbb{B} \uplus (\mathbb{N} \times \mathbb{B})$ the universe of booleans, naturals and their pairings.
Refinement $\preceq^{\mathcal{I}}$ (time shift) is the (inverse) suffix ordering

$$\frac{v \in \mathbb{D}}{v \cdot s \preceq^{\mathcal{P}} s}$$

where $v \cdot s$ is the stream $s \in \mathbb{D}^\omega$ prefixed by value $v \in \mathbb{D}$.
For instance,

$$1 \cdot (2, \mathrm{T}) \cdot \mathrm{T} \cdot \mathrm{F} \preceq^{\mathcal{I}} (2, \mathrm{T}) \cdot \mathrm{T} \cdot \mathrm{F} \preceq^{\mathcal{I}} \mathrm{T} \cdot \mathrm{F} \preceq^{\mathcal{I}} \mathrm{F} \preceq^{\mathcal{I}} \epsilon,$$

where ϵ is the empty stream.

DL Types have to be closed under time shift $\preceq$!

Typing streams (II)

Let $\textsc{Nat}^{\mathcal{I}} =_{df} \mathbb{N}^{\omega}$ and $\textsc{Bool}^{\mathcal{I}} =_{df} \mathbb{B}^{\omega}$ be ususal programming language types considered as atomic DL concepts.
Similarly $(\textsc{Nat} \times \textsc{Bool})^{\mathcal{I}} =_{df} (\mathbb{N} \times \mathbb{B})^{\omega}$.

- ϵ has no future projected behaviour, i.e. $\bot^{\mathcal{I}} = \{\epsilon\}$,
- val is a functional role, relating a stream with its first data element considered as an infinite constant stream, i.e. $val(\epsilon, \epsilon)$ and $val(v \cdot s, v^{\infty})$, e.g. $val((2, \mathtt{T}) \cdot \mathtt{T} \cdot \mathtt{F}, (2, \mathtt{T})^{\infty})$.

Typing streams (II)

Let $\text{NAT}^{\mathcal{I}} =_{df} \mathbb{N}^{\omega}$ and $\text{BOOL}^{\mathcal{I}} =_{df} \mathbb{B}^{\omega}$ be ususal programming language types considered as atomic DL concepts.
Similarly $(\text{NAT} \times \text{BOOL})^{\mathcal{I}} =_{df} (\mathbb{N} \times \mathbb{B})^{\omega}$.

- ϵ has no future projected behaviour, i.e. $\bot^{\mathcal{I}} = \{\epsilon\}$,
- val is a functional role, relating a stream with its first data element considered as an infinite constant stream, i.e. $val(\epsilon, \epsilon)$ and $val(v \cdot s, v^{\infty})$, e.g. $val((2, \mathtt{T}) \cdot \mathtt{T} \cdot \mathtt{F}, (2, \mathtt{T})^{\infty})$.

Ex. Typing streams (I): Excluded Middle

Consider stream s that starts with value 0 and turns into the infinite constant stream of Booleans T.

$$s = 0 \cdot T \cdot T \cdot T \cdot \ldots.$$

Ex. Typing streams (I): Excluded Middle

Consider stream s that starts with value 0 and turns into the infinite constant stream of Booleans T.

$$s = 0 \cdot T \cdot T \cdot T \cdot \ldots.$$

Ex. Typing streams (I): Excluded Middle

Consider stream s that starts with value 0 and turns into the infinite constant stream of Booleans T.

$$s = 0 \cdot T \cdot T \cdot T \cdot \ldots.$$

What type has s?

Ex. Typing streams (I): Excluded Middle

Consider stream s that starts with value 0 and turns into the infinite constant stream of Booleans T.

$$s = 0 \cdot T \cdot T \cdot T \cdot \ldots.$$

What type has s?

$s : \textsc{Bool} \sqcup \neg\textsc{Bool}$?

Ex. Typing streams (I): Excluded Middle

Consider stream s that starts with value 0 and turns into the infinite constant stream of Booleans T.

$$s = 0 \cdot T \cdot T \cdot T \cdot \ldots.$$

What type has s?

$s : \textsc{Bool} \sqcup \neg\textsc{Bool}$?

But: $s \notin \textsc{Bool}$ and $s \notin \neg\textsc{Bool}$

Ex. Typing streams (I): Excluded Middle

Consider stream s that starts with value 0 and turns into the infinite constant stream of Booleans T.

$$s = 0 \cdot T \cdot T \cdot T \cdot \ldots.$$

What type has s?

s : Bool $\sqcup$ $\neg$Bool?

But: $s \notin$ Bool and $s \notin \neg$Bool

excluded middle does not hold!

Ex. Typing streams (II): Disjunctive Distribution

Another classical principle does not hold: existential distribution, i.e.

$$\exists val.(C \sqcup D) \equiv \exists val.C \sqcup \exists val.D.$$

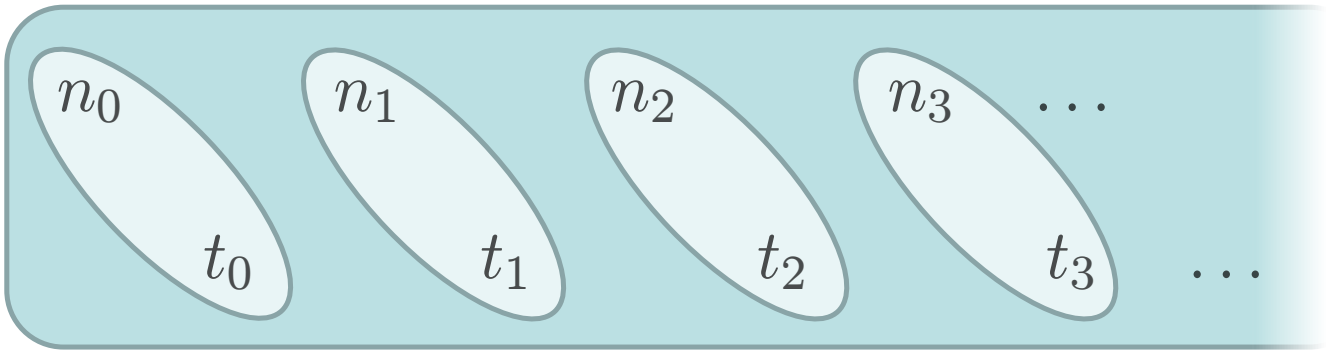
Ex. Typing streams (II): Disjunctive Distribution

table t

n_0	t_0
n_1	t_2
n_2	t_3
...	...

Ex. Typing streams (II): Disjunctive Distribution

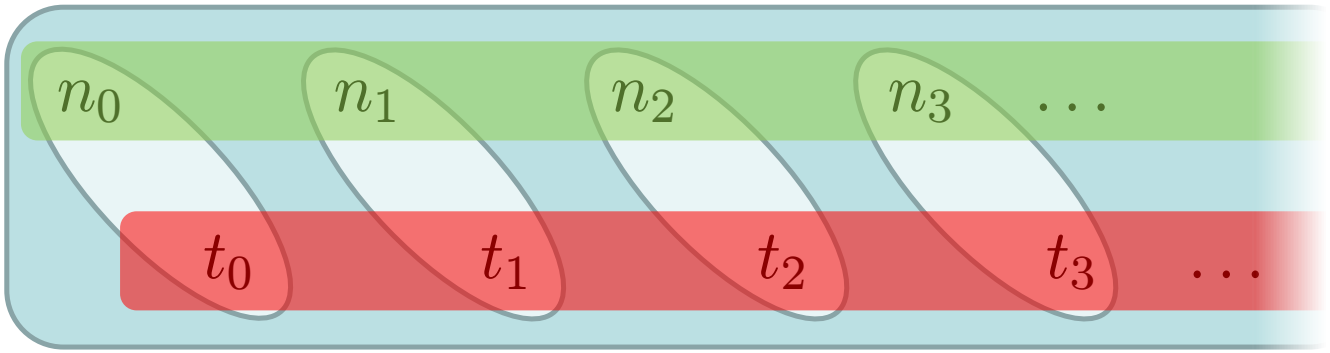
table t



$\textsc{Nat} \times \textsc{Bool}$

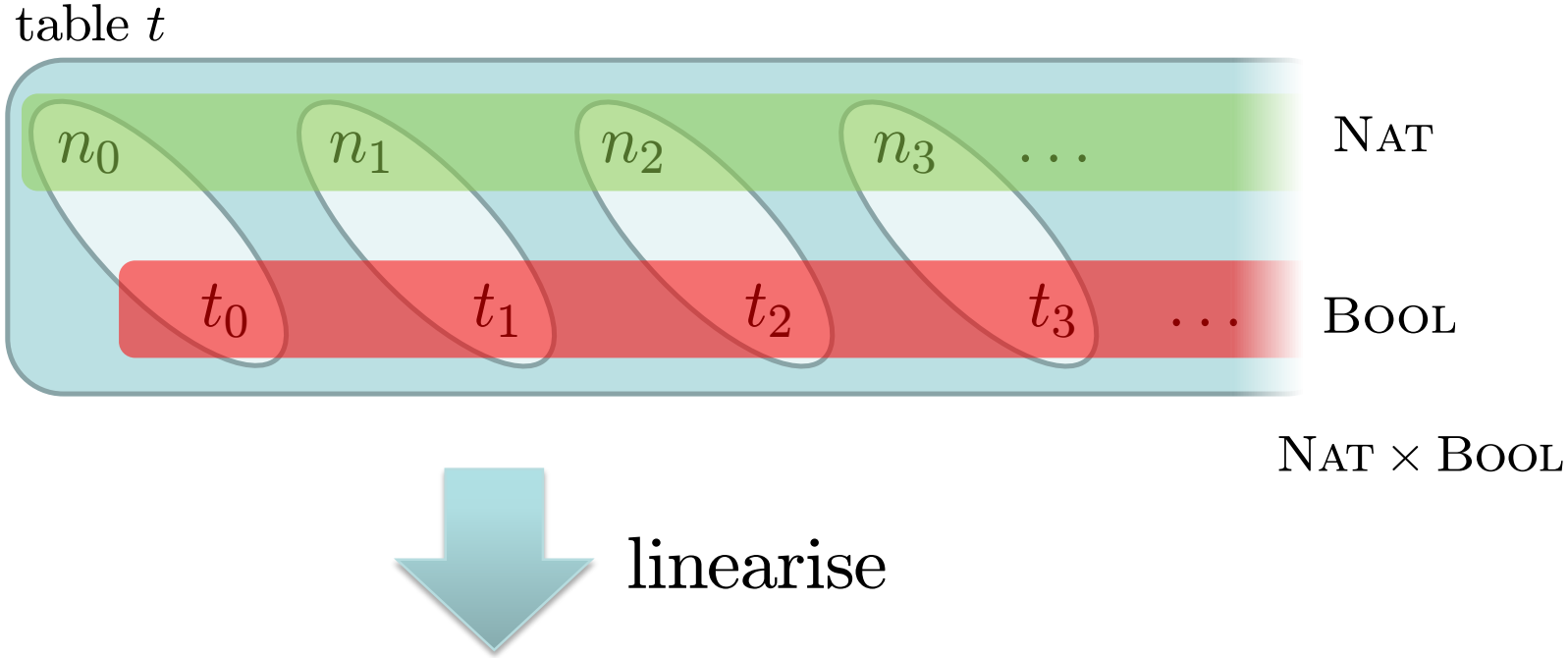
Ex. Typing streams (II): Disjunctive Distribution

table t

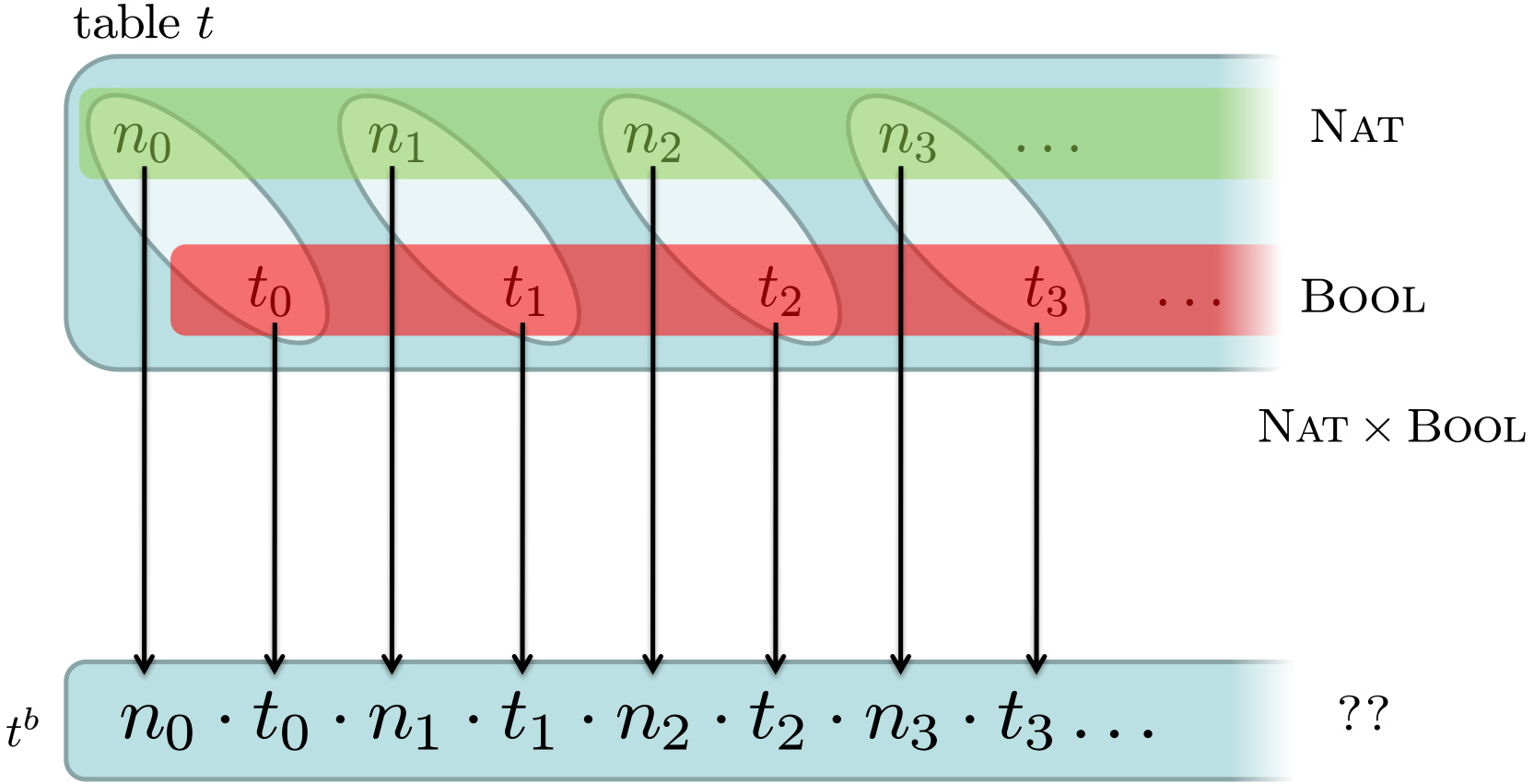


$\textsc{Nat} \times \textsc{Bool}$

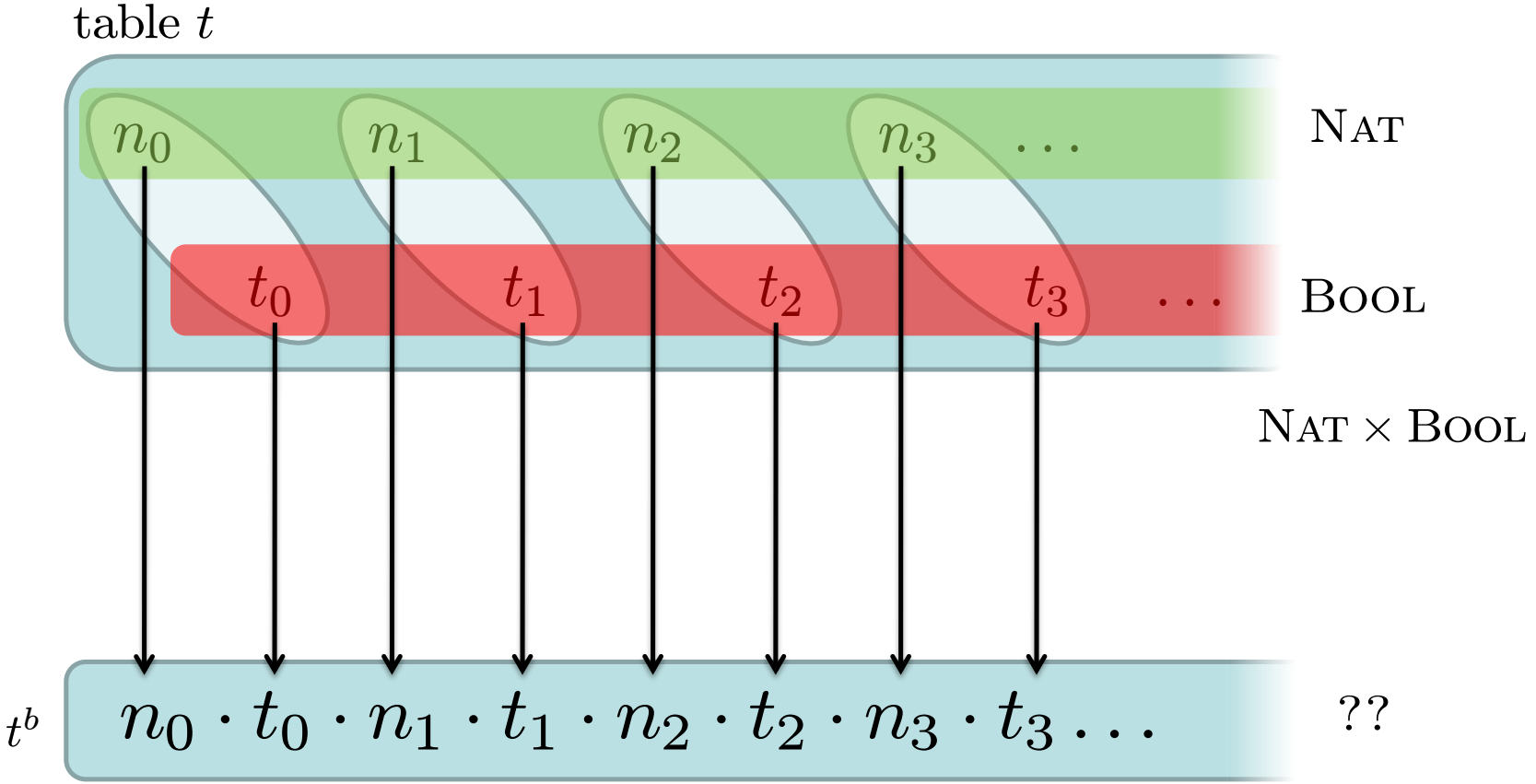
Ex. Typing streams (II): Disjunctive Distribution



Ex. Typing streams (II): Disjunctive Distribution

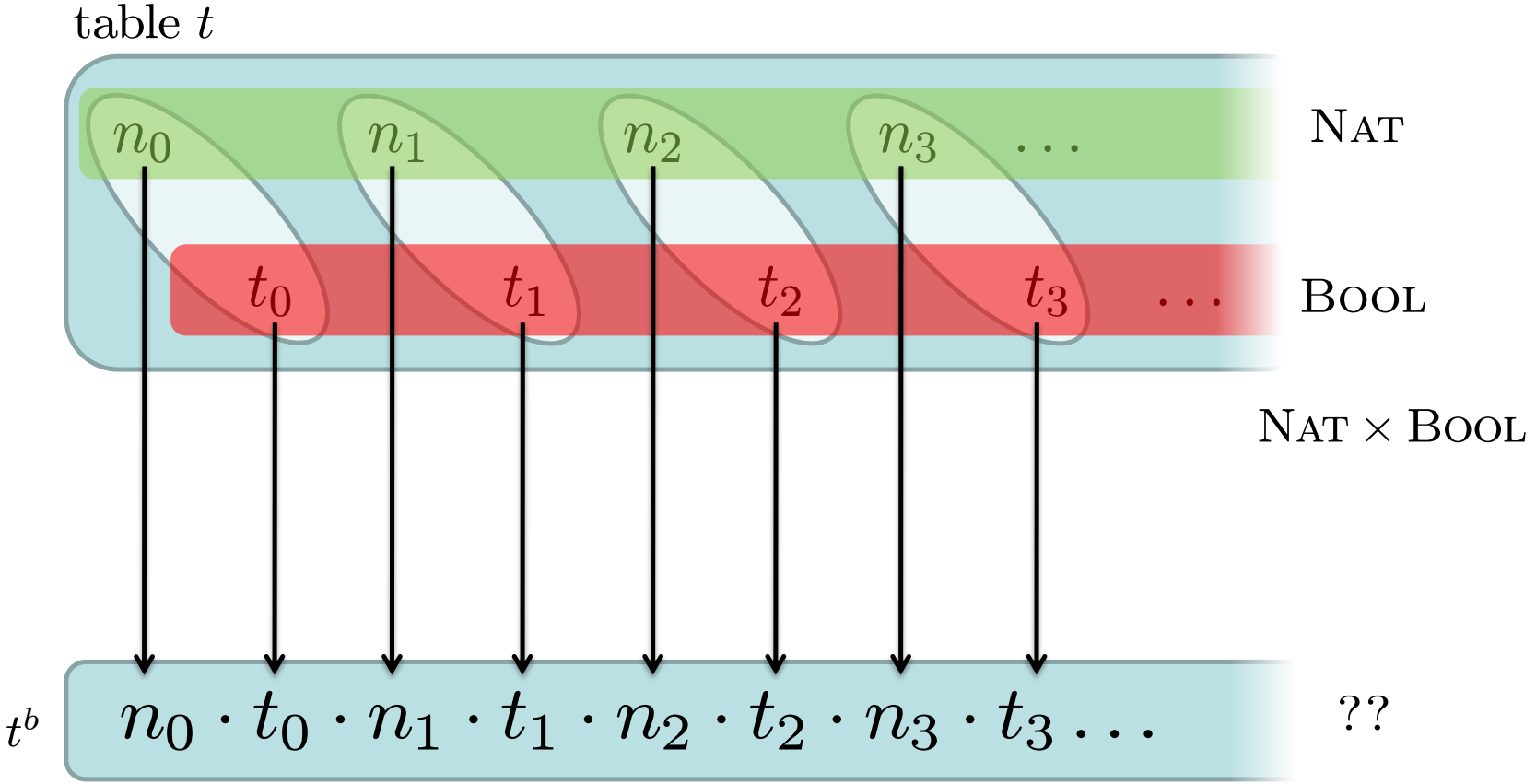


Ex. Typing streams (II): Disjunctive Distribution



What is the type of t^b?

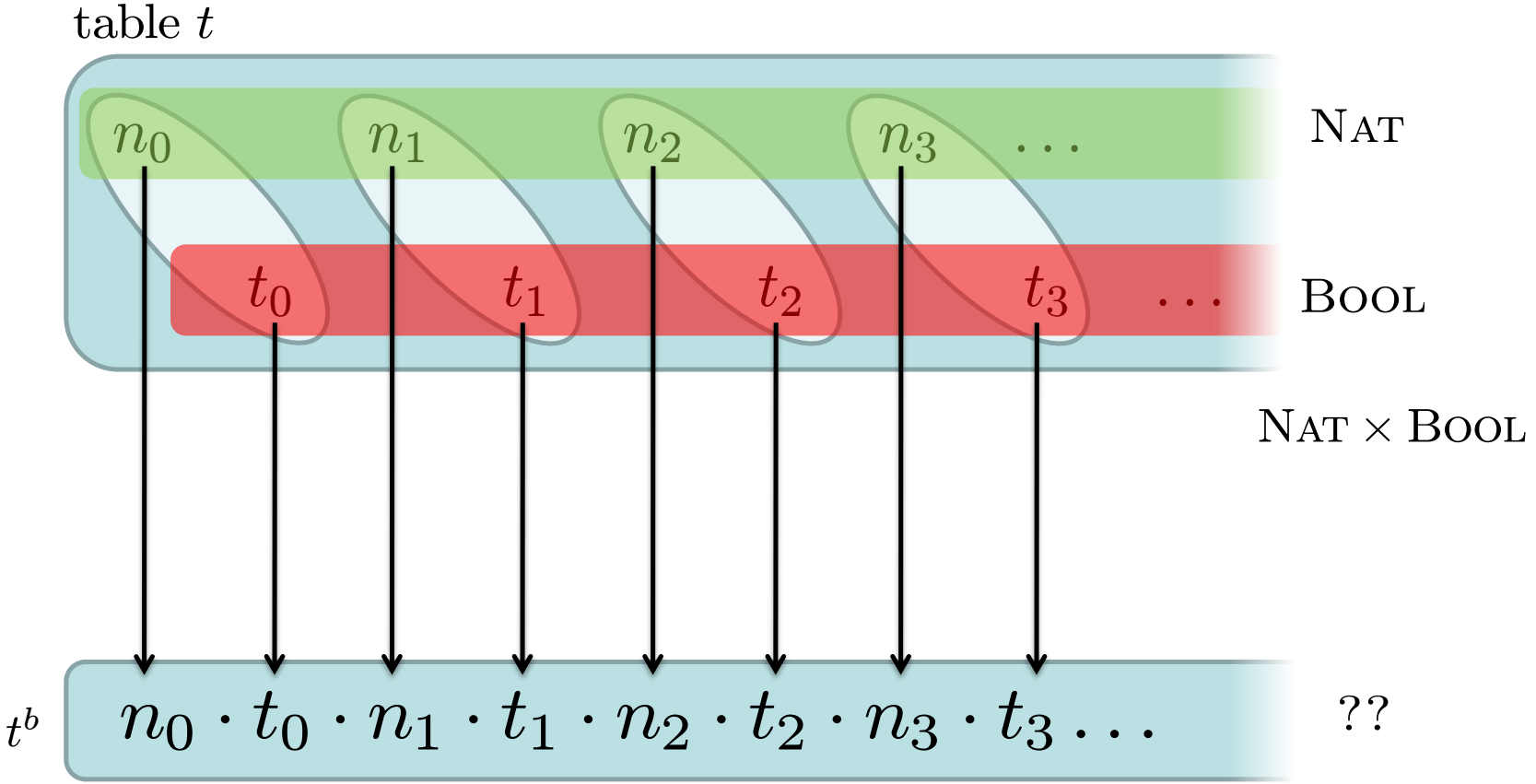
Ex. Typing streams (II): Disjunctive Distribution



What is the type of t^b?

- Nat ⊔ Bool

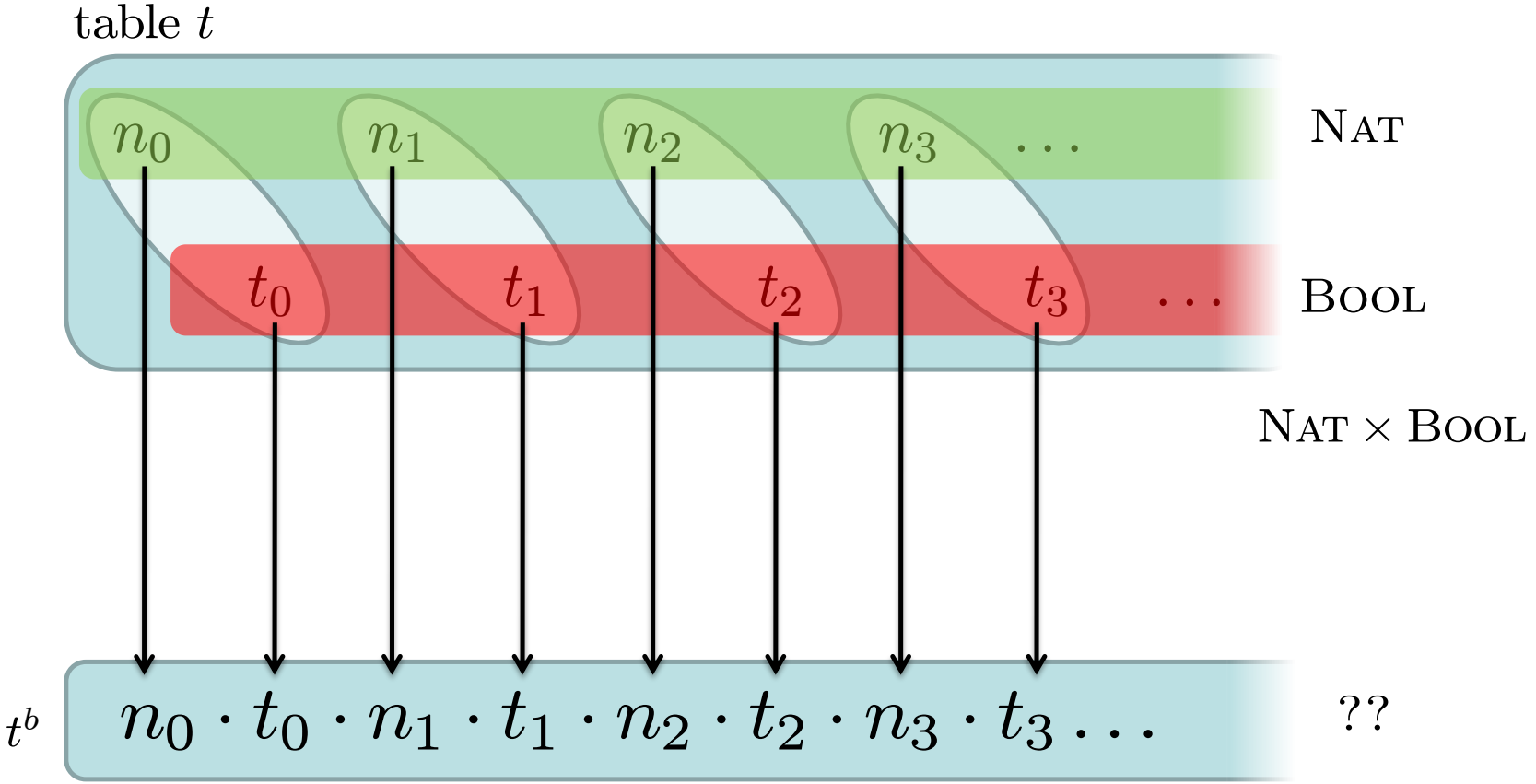
Ex. Typing streams (II): Disjunctive Distribution



What is the type of t^b?

- Nat ⊔ Bool ☒

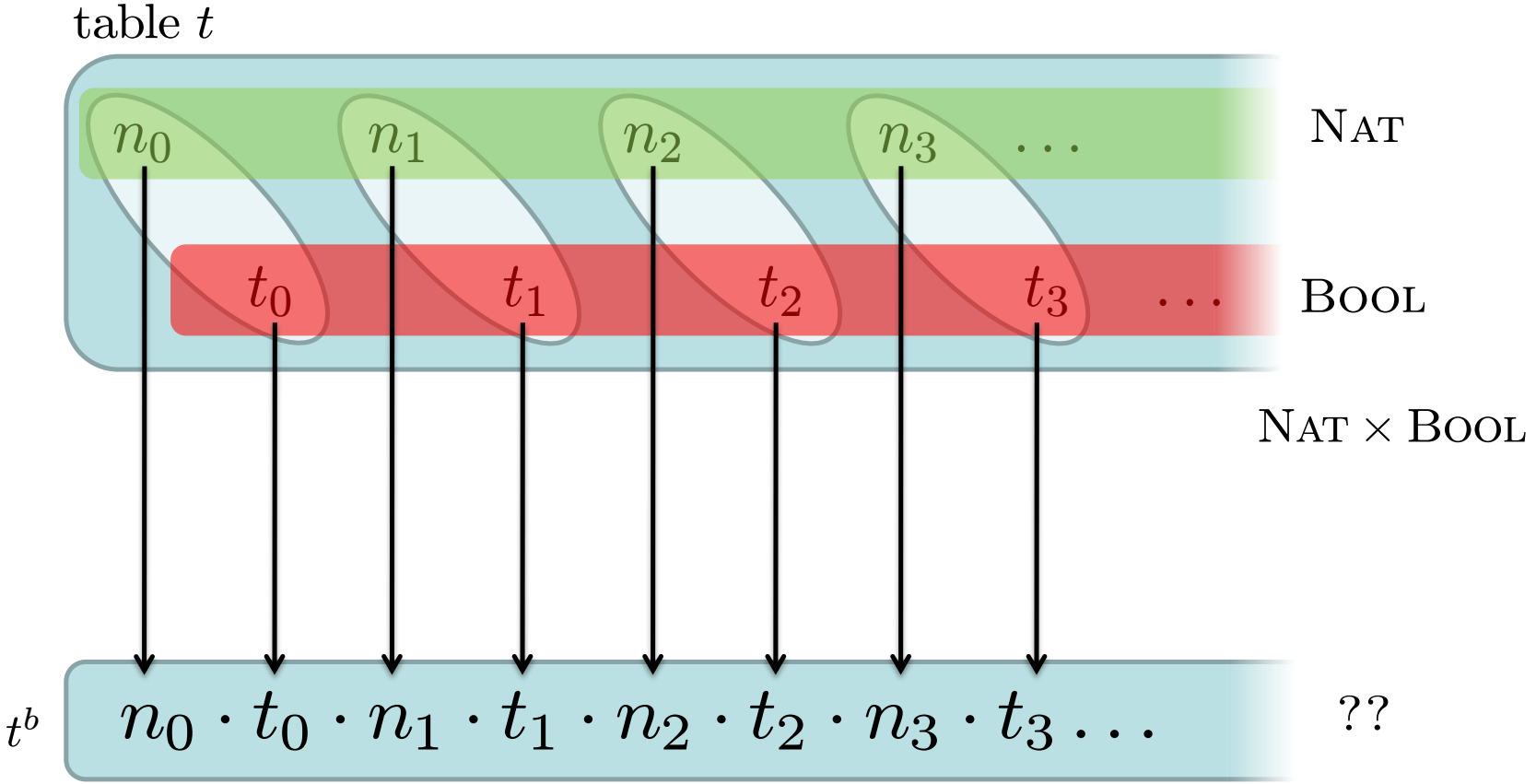
Ex. Typing streams (II): Disjunctive Distribution



What is the type of t^b?

- Nat ⊔ Bool ☒
- $\exists val.$Nat ⊔ $\exists val.$Bool

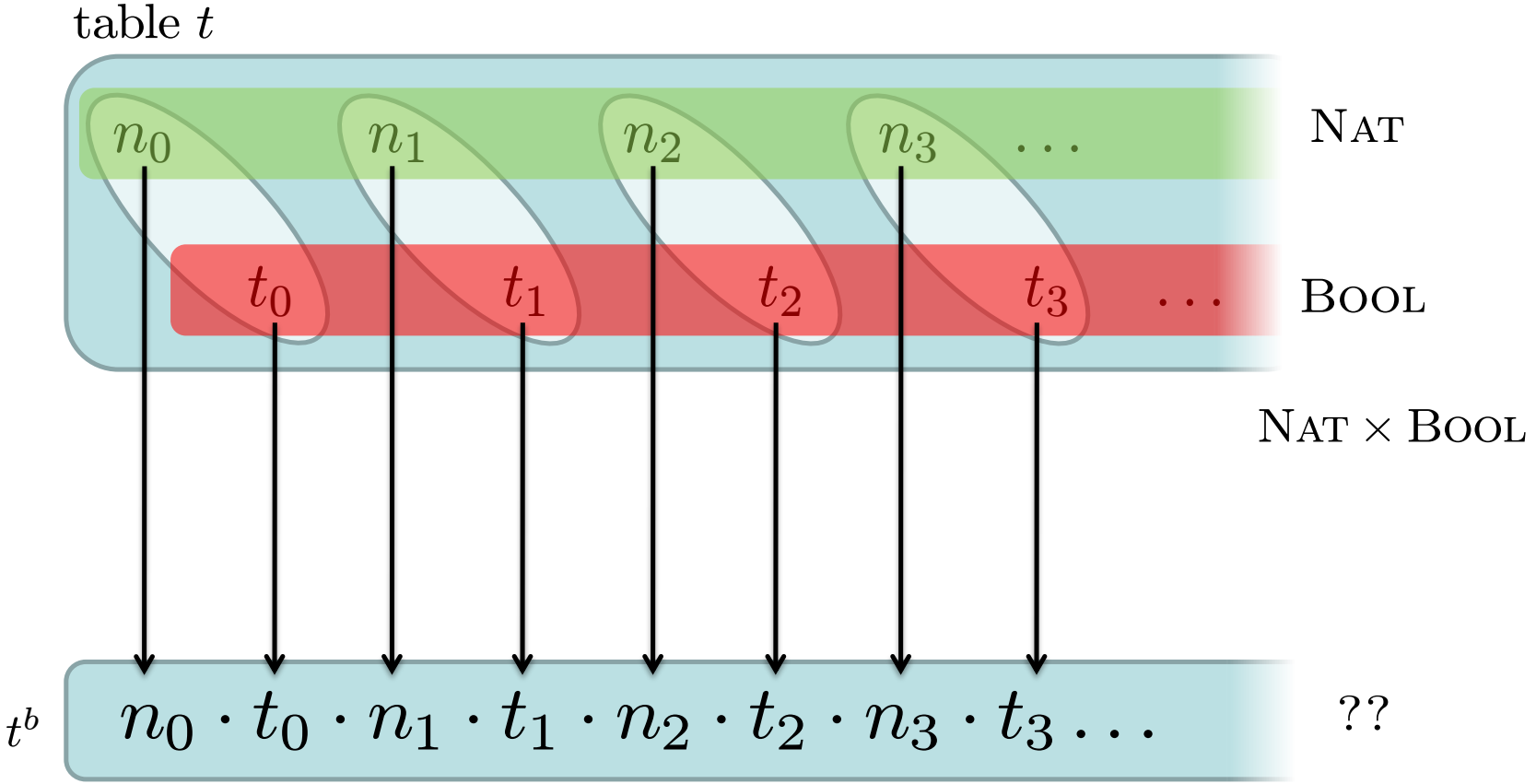
Ex. Typing streams (II): Disjunctive Distribution



What is the type of t^b?

- Nat ⊔ Bool ☒
- $\exists val.$Nat ⊔ $\exists val.$Bool ☒

Ex. Typing streams (II): Disjunctive Distribution

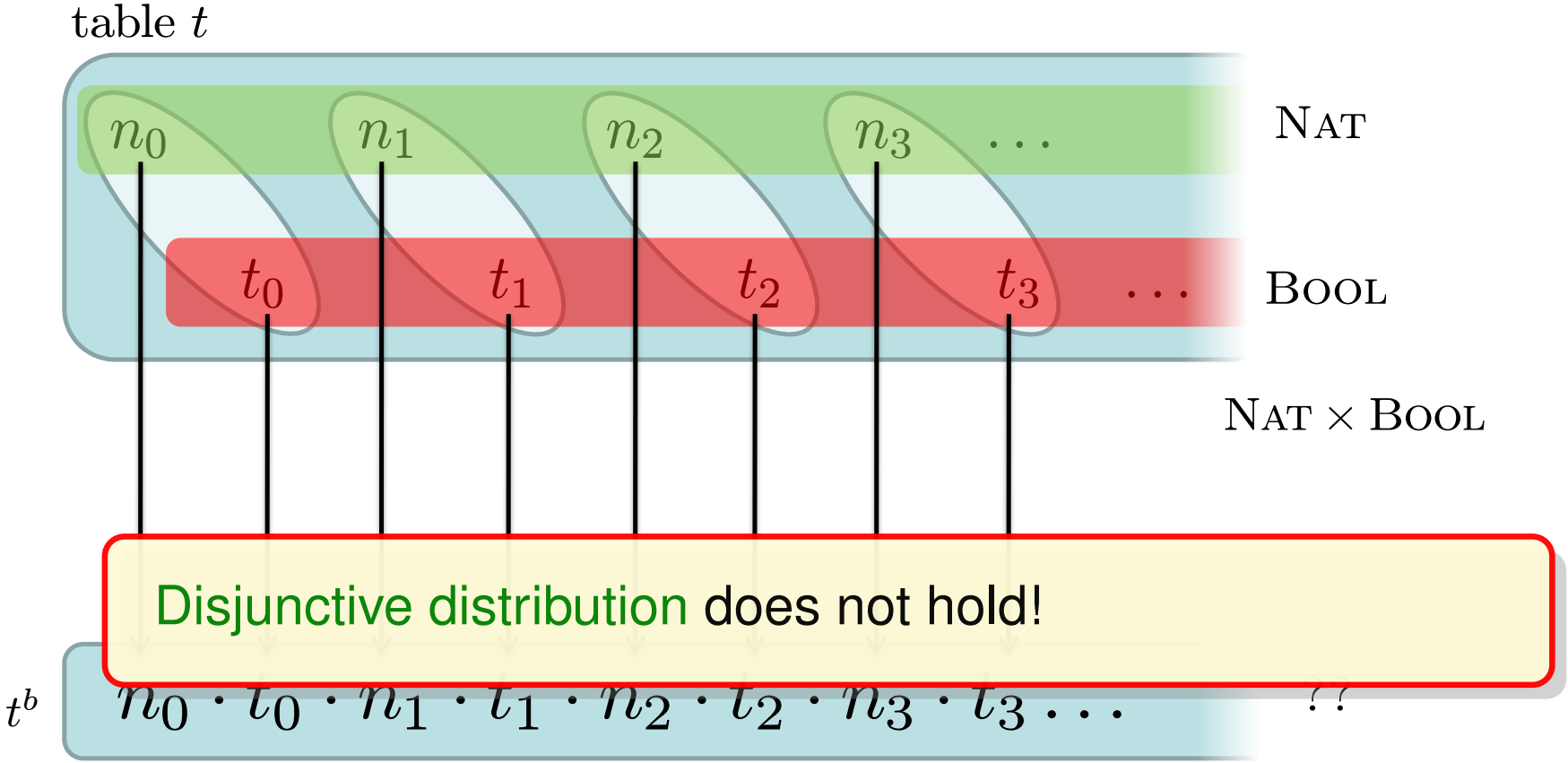


What is the type of t^b?

The correct type is NAT ∪ BOOL, expressed by

$$\exists val.(\text{NAT} \sqcup \text{BOOL})$$

Ex. Typing streams (II): Disjunctive Distribution

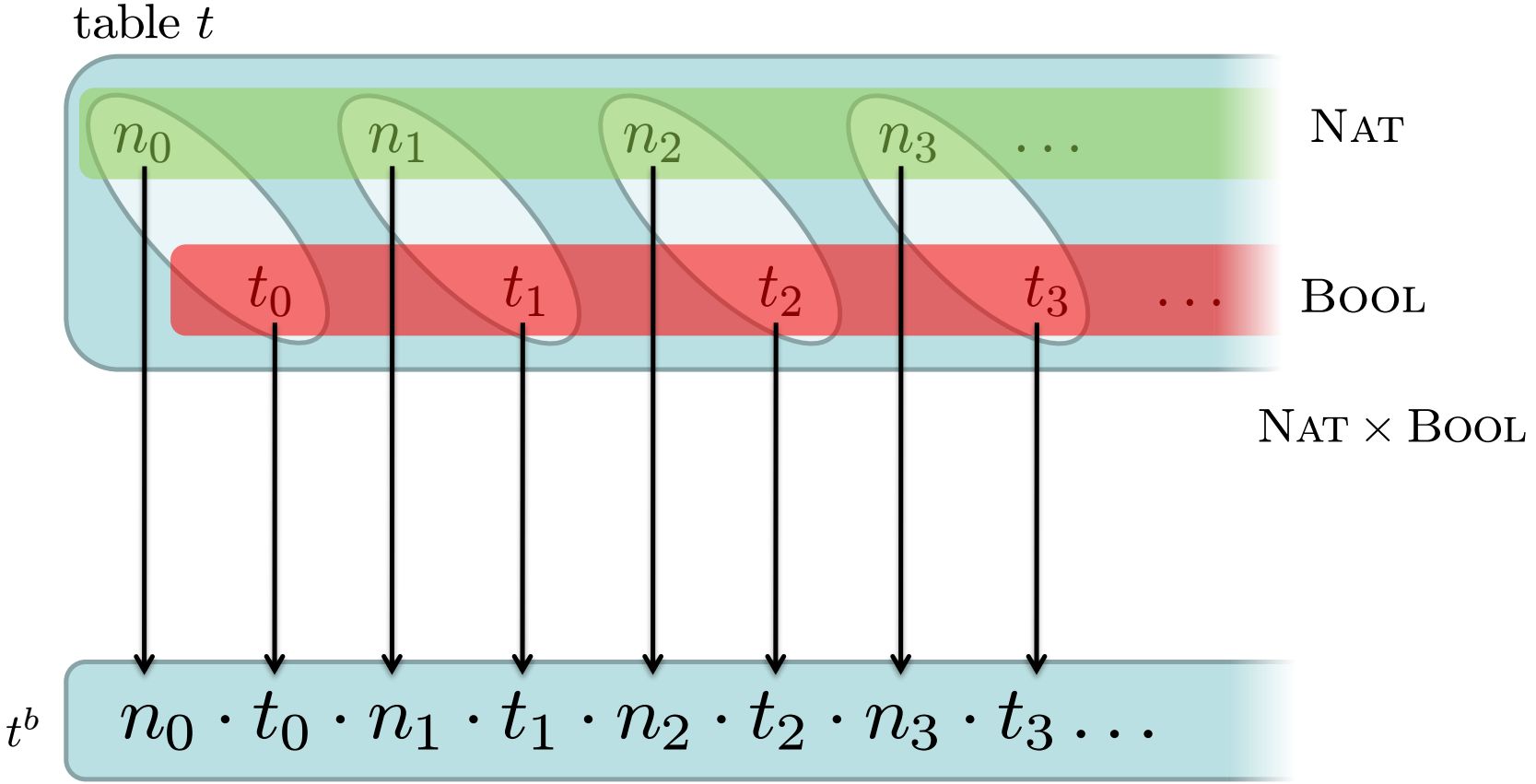


What is the type of t^b?

The correct type is Nat ∪ Bool, expressed by

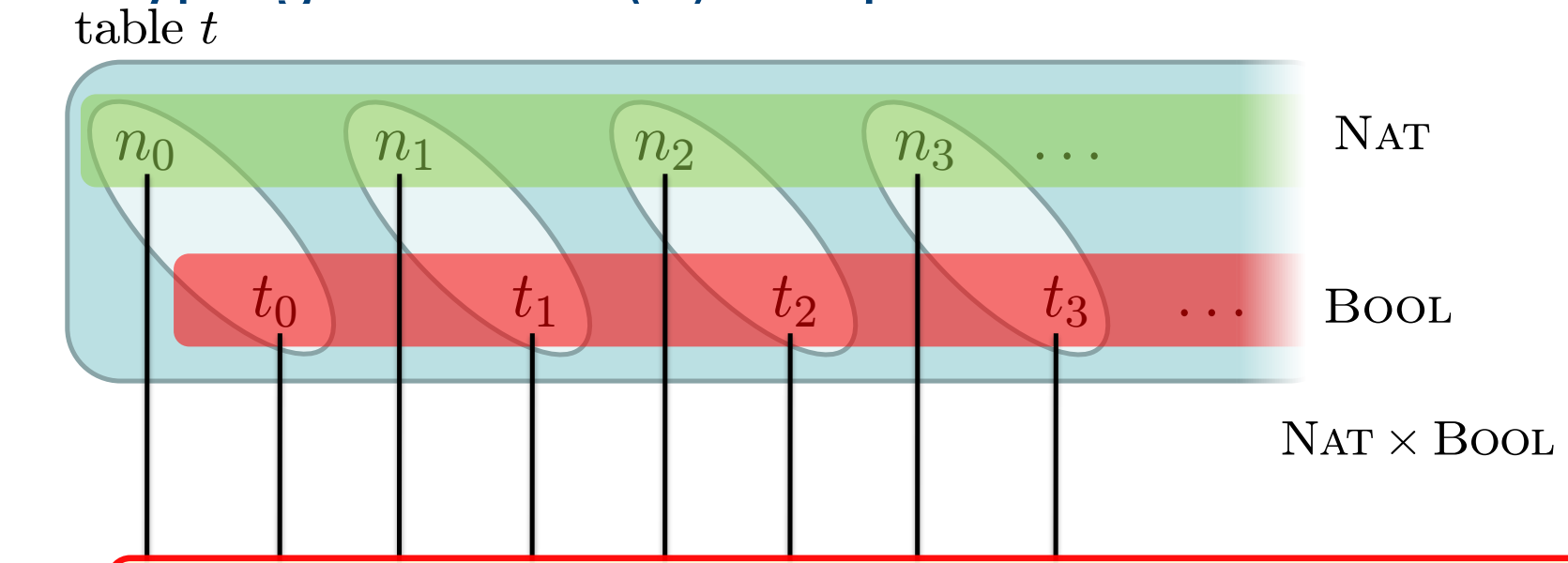
$$\exists val.(\text{Nat} \sqcup \text{Bool})$$

Ex. Typing streams (II): Disjunctive Distribution



The flattening $t \to t^b$ is a function to multiplex streams, with type $\text{Nat} \times \text{Bool} \to \exists val.(\text{Nat} \sqcup \text{Bool})$

Ex. Typing streams (II): Disjunctive Distribution



Excluded middle does not hold
Disjunctive Distribution does not hold
We need constructive semantics for Description Logic!

t^b

The flattening $t \to t^b$ is a function to multiplex streams, with type $\text{NAT} \times \text{BOOL} \to \exists val.(\text{NAT} \sqcup \text{BOOL})$

Results

We defined $c\mathcal{ALC}$ as constructive version of the Description Logic $\mathcal{ALC}$

- Syntax and Semantics for the Description Logic $c\mathcal{ALC}$
  - $\bot, C \sqcap D, C \sqcup D, \neg C, C \sqsubseteq D, \exists R.C, \forall R.C$
  - Constructive weakening of $\mathcal{ALC}$, Curry-Howard Isomorphism, suitable for typing
- Constructive Hilbert and Gentzen tableau calculi for $c\mathcal{ALC}$
- Theorems: Soundness and Completeness, Finite Model Property, Decidability
- Satisfiability of $c\mathcal{ALC}$ is PSPACE-complete

Open Problems, next steps

- Create Domain specific language (first experiments in $\mathbb{F}\#$)
- Connect stream programming language with $c\mathcal{ALC}$ typing system
- Create Audit Ontology using $c\mathcal{ALC}$
- Implement $c\mathcal{ALC}$ reasoning procedure: Type checking, type-driven compilation and optimizations

Related Work

- Constructive semantics for $\mathcal{ALC}$: Proof-theoretic vs. model-theoretic

  [?]

- Intuitionistic epistemic logic: Coding *several (partial) points of views* (different refinements)

  [?]

- Temporal DL: terminological context-dependency;

  [?, ?, ?]

- Many-Valued DL: Finitely vs. infinitely valued ($c\mathcal{ALC}$)

  [?]

- Fuzzy-DL: Use quantitative notion of approximative truth

  [?]

References I

A. Artale and E. Franconi.

A survey of temporal extensions of description logics.
Annals of Mathematics and Artificial Intelligence, 30(1–4), 2001.

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider.

The description logic handbook: theory, implementation, and applications.
Cambridge University Press, 2003.

A. Borgida.

Diachronic description logics.
In *Int'l Workshop on Description Logics (DL 2001)*, pages 106–112, 2001.

L. Botazzo, M. Ferrari, C. Fiorentini, and G. Fiorino.

A constructive semantics for ALC.
In *Int'l Workshop on Description Logics (DL 2007)*, pages 219–226, 2007.

O. Brunet.

A logic for partial system description.
Journal of Logic and Computation, 14(4):507–528, 2004.

P. F. Patel-Schneider.

A four-valued semantics for terminological logics.
Artificial Intelligence, 38:319–351, 1989.

U. Straccia.

Fuzzy ALC with fuzzy concrete domains.
In *Int'l Workshop on Description Logics (DL 2005)*, 2005.

References II

D. Nardi and R. J. Brachman.
An Introduction to Description Logics.
In *The Description Logic Handbook*, 2002.