Directions and Applications for Constructive Semantics in Description Logics

Loris Bozzato

DKM - Data and Knowledge Management Research Unit,
FBK-Irst, Fondazione Bruno Kessler - Trento, Italy

DKM internal seminar
May 2011
Acknowledgements

This presentation is a summary of my work with:

- **Mauro Ferrari**
  DICOM, Università degli Studi dell’Insubria - Varese, Italy

- **Camillo Fiorentini**
  DSI, Università degli Studi di Milano - Milano, Italy

- **Guido Fiorino**
  DIMEQUANT, Università degli Studi di Milano-Bicocca - Milano, Italy

- **Paola Villa**
  PhD at DICOM, Università degli Studi dell’Insubria - Varese, Italy
Outline

1. Introduction
2. *BCDL*: information terms semantics for *ALC*
3. *KALC*\(\infty\): Kripke-style semantics for *ALC*
4. *KALC*: decidable Kripke-style interpretation for *ALC*
5. Application directions
Introduction

1. **Introduction**

2. **BCDL**: information terms semantics for $\mathcal{ALC}$

3. **$\mathcal{KALC}^\infty$**: Kripke-style semantics for $\mathcal{ALC}$

4. **$\mathcal{KALC}$**: decidable Kripke-style interpretation for $\mathcal{ALC}$

5. Application directions
Constructive logics

Formalizations of ideas from constructivism

A proof of \( A \land B \) is composed from proofs of \( A \) and \( B \).

A proof of \( A \lor B \) is composed of a proof for \( A \) or \( B \).

A proof of \( A \rightarrow B \) is construction transforming proofs of \( A \) in proofs for \( B \).

\( \bot \) is an unprovable formula (thus \( A \rightarrow \bot = \neg A \)).

Possible formalizations:
- Intuitionism
- Recursive realizability
- Information terms semantics
Constructive logics

Formalizations of ideas from constructivism

Brouwer-Heyting-Kolmogorov (BHK) or proof interpretation

Semiformal presentation of a constructive semantics

E.g. propositional part:

- A proof of $A \land B$ is composed from proofs of $A$ and $B$
- A proof of $A \lor B$ is composed of a proof for $A$ or $B$
- A proof of $A \rightarrow B$ is construction transforming proofs of $A$ in proofs for $B$
- $\bot$ is an unprovable formula (thus $A \rightarrow \bot = \neg A$)

Possible formalizations:

- Intuitionism
- Recursive realizability
- Information terms semantics
Constructive logics

- **Characteristic properties:**
  - **Disjunction property** (DP):
    “Whenever it proves a disjunction formula, it proves one of the disjoints”
  - **Explicit definability property** (ED):
    “Whenever it proves an existential formula, it presents a witness of the existence”

- **Constructivism and Computer Science:**
  - **Formulas-as-types** (Curry-Howard isomorphism)
  - **Proofs-as-programs**
Constructive description logics

Constructive interpretations of description logics

Motivations

- Computational interpretation of proofs and formulas
- Useful in domains with dynamic and incomplete knowledge

Known proposals

- [de Paiva, 2005]: translations of DLs in constructive systems
- [Kaneiwa, 2005]: definitions for different constructive negations in DLs
- [Odintsov and Wansing, 2003]: inconsistency tolerant version of DLs
- [Mendler and Scheele, 2010]: Kripke semantics with “fallible” elements
Introduction | Constructive description logics

Proposal [de Paiva, 2005]

Motivation

- Extend proof-theoretical results (Curry-Howard) on DLs
- Define a context-sensitive DL

Proposals

3 different interpretations of $\mathcal{ALC}$ in constructive systems:

- $\text{IALC}$: from $\mathcal{ALC}$ to IFOL (via $\mathcal{ALC} \rightarrow \text{FOL}$ translation)
- $\text{iALC}$: from $\mathcal{ALC}$ to IK (via $\mathcal{ALC} \rightarrow K_m$ translation)
- $\text{cALC}$: from $\mathcal{ALC}$ to CK (via $\mathcal{ALC} \rightarrow K_m$ translation)
Proposal [Kaneiwa, 2005]

Motivation

Representation of different notions of negative information in DLs (contraries, contradictories and subcontraries)

E.g. difference between $\text{Happy}$, $\text{Unhappy}$, $\neg \text{Happy}$, $\neg \text{Unhappy}$

Proposals

- 2 different extensions to $\mathcal{ALC}$ semantics
  (different interactions between constructive and classical negation)
- tableaux algorithm for satisfiability

Similar works: [Kamide, 2010a, Kamide, 2010b]

Paraconsistent and temporal versions of $\mathcal{ALC}$, based on a similar semantics
Proposal [Odintsov and Wansing, 2003]

**Ideas**

- Paraconsistent versions of $\mathcal{ALC}$
- Constructive semantics to represent partial information

**Proposal [Odintsov and Wansing, 2003]**

- 3 constructive paraconsistent semantics for $\mathcal{ALC}$
  (Different translations to four valued logic $N4$)
- complete tableaux calculus for each logic

**Further work** [Odintsov and Wansing, 2008]

Reviews of calculi, tableaux procedure for one of the presented logics
Proposal [Mendler and Scheele, 2010]

Idea

- Representation of partial knowledge and consistency under abstraction
- Evolving OWA: stages of information with changing properties and abstract individuals

Proposal

$c\text{ALC}$: Kripke semantics for $\text{ALC}$ with fallible entities

- fallible entities $\bot^T$: contradictory domain elements (maximal poset elements or undefined role fillers)
- complete and decidable Hilbert and tableaux calculi

Application [Mendler and Scheele, 2009]

Reasoning on data streams in auditing domain
Our proposals

**BCDL** [Bozzato et al., 2007, Bozzato et al., 2009b, Ferrari et al., 2010]

- Information terms semantics + natural deduction calculus
  - computational interpretation of proofs (*Proofs as programs*)

**K\(\mathcal{ALC}^\infty\)** [Bozzato et al., 2009a, Villa, 2010]

- Kripke-style semantics + tableaux calculus
  - possibly infinite models, efficient treatment of implications

**K\(\mathcal{ALC}\)** [Bozzato et al., 2010, Bozzato, 2011]

- Kripke-style semantics + tableaux algorithm
  - finite models, decidability from terminating tableau procedure
Outline

1. Introduction

2. \textit{BCDL}: information terms semantics for \textit{ALC}

3. \textit{KALC}^\infty: Kripke-style semantics for \textit{ALC}

4. \textit{KALC}: decidable Kripke-style interpretation for \textit{ALC}

5. Application directions
**BCDL**: Basic Constructive Description Logic [Ferrari et al., 2010]

- Information terms semantics for $\mathcal{ALC}$
- Natural deduction calculus $\mathcal{ND}_c$

**Information terms** [Miglioli et al., 1989]

Syntactic objects justifying validity of formulas in classical models

- realization of BHK interpretation
- related to realizability interpretations
**BCDL**: information terms semantics for **ALC**

**BCDL**: Basic Constructive Description Logic [Ferrari et al., 2010]
- Information terms semantics for **ALC**
- Natural deduction calculus $\mathcal{ND}_c$

Information terms [Miglioli et al., 1989]
- Syntactic objects justifying validity of formulas in classical models
  - realization of BHK interpretation
  - related to realizability interpretations

**Features:**
- Classical reading of DL formulas
- Simple proof theoretical characterization by $\mathcal{ND}_c$
- Computational interpretation of proofs
- Natural notion of state
ALCG syntax and classical semantics

Syntax is the same as $\mathcal{ALC}$, adding a set of generators $\mathcal{NG}$

**Concepts**

$$C ::= A \mid G \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists R.C \mid \forall R.C$$

**Formulas**

$$K ::= \bot \mid (t,s) : R \mid t : C \mid \forall G C$$

where $A \in \mathcal{NC}$, $R \in \mathcal{NR}$, $t,s \in \mathcal{NI}$ and $G \in \mathcal{NG}$

**Generators $\mathcal{NG}$**

Concepts over fixed set of individual names $dom(G) = \{c_1, \ldots, c_n\}$

→ limited form of subsumption: $\forall_G C \equiv G \sqsubseteq C$
Validity of formulas: given a model $\mathcal{M} = (\Delta^\mathcal{M}, \cdot^\mathcal{M})$

$\mathcal{M} \not\models \bot$

$\mathcal{M} \models (t, s) : R \iff (t^\mathcal{M}, s^\mathcal{M}) \in R^\mathcal{M}$

$\mathcal{M} \models t : H \iff t^\mathcal{M} \in H^\mathcal{M}$

$\mathcal{M} \models \forall_G H \iff G^\mathcal{M} = \{c_1^\mathcal{M}, \ldots, c_n^\mathcal{M}\} \subseteq H^\mathcal{M}$
Information terms $\text{IT}_N(K)$

Structured objects that **constructively justify the truth** of a formula $K$

- $\text{IT}_N(K) = \{tt\}$, if $K$ is atomic
- $\text{IT}_N(c : C_1 \sqcup C_2) = \{(k, \alpha) \mid k \in \{1, 2\} \text{ and } \alpha \in \text{IT}(c : C_k)\}$
**BCDL** information terms semantics

**Information terms** $\text{IT}_N(K)$

Structured objects that **constructively justify the truth** of a formula $K$

\[
\text{IT}_N(K) = \{ tt \}, \text{ if } K \text{ is atomic}
\]

\[
\text{IT}_N(c : C_1 \sqcup C_2) = \{ (k, \alpha) | k \in \{1, 2\} \text{ and } \alpha \in \text{IT}(c : C_k) \}
\]

**Realizability** $\mathcal{M} \triangleright \langle \alpha \rangle K$

Truth of $K$ in a model $\mathcal{M}$ justified w.r.t. $\alpha$

\[
\mathcal{M} \triangleright \langle tt \rangle K \text{ iff } \mathcal{M} \models K
\]

\[
\mathcal{M} \triangleright \langle (k, \alpha) \rangle c : C_1 \sqcup C_2 \text{ iff } \mathcal{M} \triangleright \langle \alpha \rangle c : C_k
\]
BCDL information terms semantics

Information terms $\text{IT}_N(K)$
Structured objects that **constructively justify the truth** of a formula $K$

$\text{IT}_N(K) = \{tt\}$, if $K$ is atomic
$\text{IT}_N(c : C_1 \sqcup C_2) = \{(k, \alpha) \mid k \in \{1, 2\} \text{ and } \alpha \in \text{IT}(c : C_k)\}$

Realizability $\mathcal{M} \triangleright \langle \alpha \rangle K$
Truth of $K$ in a model $\mathcal{M}$ justified w.r.t. $\alpha$

$\mathcal{M} \triangleright \langle tt \rangle K$ iff $\mathcal{M} \models K$
$\mathcal{M} \triangleright \langle (k, \alpha) \rangle c : C_1 \sqcup C_2$ iff $\mathcal{M} \triangleright \langle \alpha \rangle c : C_k$

Theorem (classical and IT semantics)
$\mathcal{M} \models K$ iff there exists $\alpha \in \text{IT}(K)$ such that $\mathcal{M} \triangleright \langle \alpha \rangle K$
$\text{IT}_N(K) = \{tt\}$ for $K$ atomic or negated

$\text{IT}_N(c : C_1 \cap C_2) = \text{IT}(c : C_1) \times \text{IT}(c : C_2)$

$\text{IT}_N(c : C_1 \cup C_2) = \text{IT}(c : C_1) \uplus \text{IT}(c : C_2)$

$\text{IT}_N(c : \exists R.C) = N \times \bigcup_{d \in N} \text{IT}(d : C)$

$\text{IT}_N(c : \forall R.C) = (\bigcup_{d \in N} \text{IT}(d : C))^N$

$\text{IT}_N(\forall G) = \left( \bigcup_{d \in \text{dom}(G)} \text{IT}(d : C) \right)^{\text{dom}(G)}$

$\mathcal{M} \models \langle \text{true} \rangle K$ iff $\mathcal{M} \models K$

$\mathcal{M} \models \langle (\alpha, \beta) \rangle c : C_1 \cap C_2$ iff $\mathcal{M} \models \langle \alpha \rangle c : C_1$ and $\mathcal{M} \models \langle \beta \rangle c : C_2$

$\mathcal{M} \models \langle (k, \alpha) \rangle c : C_1 \cup C_2$ iff $\mathcal{M} \models \langle \alpha \rangle c : C_k$

$\mathcal{M} \models \langle (d, \alpha) \rangle c : \exists R.C$ iff $\mathcal{M} \models (c, d) : R$ and $\mathcal{M} \models \langle \alpha \rangle d : C$

$\mathcal{M} \models \langle \phi \rangle c : \forall R.C$ iff $\mathcal{M} \models c : \forall R.C$ and, for every $d \in N$, if $\mathcal{M} \models (c, d) : R$ then $\mathcal{M} \models \langle \phi(d) \rangle d : C$

$\mathcal{M} \models \langle \phi \rangle \forall G$ iff, for every $d \in \text{dom}(G)$, $\mathcal{M} \models \langle \phi(d) \rangle d : C$
Calculus $\mathcal{ND}$: natural deduction calculus for $\mathcal{ALCG}$

Rules

\[
\begin{align*}
\Gamma & \vdash \pi' \\
\top : A_k \\
t : A_1 \sqcup A_2 & \vdash I_k \\
\Gamma_1 \vdash \pi_1 \\
\Gamma_2, [(t, p) : R, p : A] & \vdash \pi_2 \\
t : \exists R. A & \vdash K \\
\Gamma' & \vdash \pi' \\
\forall G A & \vdash t : G \\
\forall G E & \vdash t : A \\
\top & \vdash \perp \\
\Gamma, [t : \neg H] & \vdash \neg E
\end{align*}
\]

Theorem

- $\mathcal{ND}$ is sound and complete w.r.t. $\mathcal{ALCG}$
- Leaving aside generators rules, $\mathcal{ND}$ is sound and complete w.r.t. $\mathcal{ALC}$
Calculus $\mathcal{ND}_c$: natural deduction calculus for $BCDL$

Rules

Every rule from $\mathcal{ND}$ (minus $\neg E$)

\[
\frac{\Gamma \vdash \pi'}{t : \neg \neg C} \quad \text{At} \quad \frac{\Gamma \vdash \pi'}{t : \forall R. \neg \neg H} \quad \text{KUR}
\]

\[t : \neg \neg \forall R. H \]

Note

KUR corresponds to the Kuroda axiom schema Kur

\[\text{Kur} \equiv \forall x. \neg \neg A(x) \rightarrow \neg \neg \forall x. A(x)\]
Soundness of $\mathcal{ND}_c$

**Operator $\Phi_{\mathcal{N}}^\pi$:** Given a proof $\pi : \Gamma \vdash K$ over $\mathcal{N}$:

$$\Phi_{\mathcal{N}}^\pi : IT_{\mathcal{N}}(\Gamma) \rightarrow IT_{\mathcal{N}}(K)$$

**Note:** computable function, inductively defined on depth of $\pi$

---

**Example: $\sqcup I_k$**

If last rule in $\pi$ is $\sqcup I_k$ with $k \in \{1, 2\}$, then:

$$\Phi_{\mathcal{N}}^\pi : IT_{\mathcal{N}}(\Gamma) \rightarrow IT_{\mathcal{N}}(t : C_1 \sqcup C_2)$$

defined as:

$$\Phi_{\mathcal{N}}^\pi(\overline{\gamma}) = (k, \Phi_{\mathcal{N}}^{\pi'}(\overline{\gamma}))$$

---

**$\sqcup I_k$ rule**

$$
\begin{array}{c}
\Gamma \\
\vdots \pi' \\
t : A_k \\
\hline
\sqcup I_k \\
\hline
t : A_1 \sqcup A_2
\end{array}
$$
Soundness of $\mathcal{ND}_c$: computational interpretation

**Theorem (Soundness)**

If $\pi : \Gamma \vdash K$, then:

- $\Gamma \models K$.
- If $M \triangleright \langle \gamma \rangle \Gamma$ then $M \triangleright \langle \Phi^\pi_N(\gamma) \rangle K$. (constructive consequence)

**Computational interpretation**

- Given a proof $\pi$ of a formula $K$...
- ...its $\Phi^\pi_N$ provides a “program” to compute an IT for $K$
Theorem (Completeness)

\( \Gamma \vdash_{BCDL} K \) iff \( K \) is a constructive consequence of \( \Gamma \).

Constructive properties

For \( \Gamma \) of Harrop formulas (no \( \sqcap \) and \( \exists \)):

- **Disjunction property (DP):**
  If \( \Gamma \vdash_{BCDL} c : A \sqcup B \), then \( \Gamma \vdash_{BCDL} c : A \) or \( \Gamma \vdash_{BCDL} c : B \).

- **Explicit definability property (EDP):**
  If \( \Gamma \vdash_{BCDL} c : \exists R.A \), then there exists \( d \in NI \) such that \( \Gamma \vdash_{BCDL} (c, d) : R \) and \( \Gamma \vdash_{BCDL} d : A \).
1. Introduction

2. $BCDL$: information terms semantics for $ALC$

3. $KALC^\infty$: Kripke-style semantics for $ALC$

4. $KALC$: decidable Kripke-style interpretation for $ALC$

5. Application directions
\( \mathcal{KALC}^\infty \): Kripke-style semantics for \( \mathcal{ALC} \)

- Kripke-style semantics for \( \mathcal{ALC} \)
- Tableaux calculus \( \mathcal{T} \)

**Features:**
- Final state conditions on models, related to Kur
- Tableaux calculus with efficient treatment of duplications
$\mathcal{KALC}^{\infty}$ syntax and classical semantics

### Concepts

$$C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid C \rightarrow C \mid \exists R.C \mid \forall R.C$$

### Formulas

$$K ::= \bot \mid (t,s) : R \mid t : C \mid C$$

where $A \in NC$, $R \in NR$, $t,s \in NI$.

→ compared to $\mathcal{BCDL}$: implication (unrestricted $\sqsubseteq$) and no generators
$\mathcal{KALC}^\infty$: Kripke-style semantics for $\mathcal{ALC}$

### Concepts

\[ C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid C \rightarrow C \mid \exists R.C \mid \forall R.C \]

### Formulas

\[ K ::= \bot \mid (t,s) : R \mid t : C \mid C \]

where $A \in \text{NC}$, $R \in \text{NR}$, $t,s \in \text{NI}$.

→ compared to $\text{BCDL}$: implication (unrestricted $\sqsubseteq$) and no generators

### Validity of formulas

Given a model $\mathcal{M} = (\Delta^\mathcal{M}, \cdot^\mathcal{M})$

- $\mathcal{M} \not\models \bot$
- $\mathcal{M} \models (t,s) : R$ if $$(t^\mathcal{M}, s^\mathcal{M}) \in R^\mathcal{M}$$
- $\mathcal{M} \models t : C$ if $t^\mathcal{M} \in C^\mathcal{M}$
- $\mathcal{M} \models C$ if $C^\mathcal{M} = \Delta^\mathcal{M}$
K\textit{ALC}^\infty: Kripke-style semantics for ALC

**Rooted poset:**

Given a (rooted) poset \((P, \leq)\), **final**: \(\phi \in P\) such that, for every \(\alpha \in P\), \(\phi \leq \alpha\) implies \(\phi = \alpha\).

**K-posets**

- Given a (rooted) poset \((P, \leq)\), **final**: \(\phi \in P\) such that, for every \(\alpha \in P\), \(\phi \leq \alpha\) implies \(\phi = \alpha\).
- **K-poset**: every poset \((P, \leq)\) such that, for every \(\alpha \in P\), \(\text{Fin}(\alpha) \neq \emptyset\).
KALC\(^\infty\)-models

**KALC\(^\infty\)-model** \(K = \langle P, \leq, \rho, \iota \rangle\)

- \((P, \leq)\) is a K-poset with root \(\rho\);
\( \mathcal{KALC}^\infty \)-models

\( \mathcal{KALC}^\infty \)-model \( K = \langle P, \leq, \rho, i \rangle \)

- \((P, \leq)\) is a \( K \)-poset with root \( \rho \);
- \( i(\alpha) = (D^\alpha, \cdot^\alpha) \) such that, for every \( \alpha, \beta \in P \) with \( \alpha \leq \beta \):
  1. \( D^\alpha \subseteq D^\beta \);
  2. for every \( c \in NI \), \( c^\alpha = c^\beta \);
  3. for every \( C \in NC \), \( C^\alpha \subseteq C^\beta \);
  4. for every \( R \in NR \), \( R^\alpha \subseteq R^\beta \).
$\mathcal{KALC}^\infty$-models

$\mathcal{KALC}^\infty$-model $K = \langle P, \leq, \rho, \iota \rangle$

- $(P, \leq)$ is a $K$-poset with root $\rho$;
- $\iota(\alpha) = (D^\alpha, \cdot^\alpha)$ such that, for every $\alpha, \beta \in P$ with $\alpha \leq \beta$:
  1. $D^\alpha \subseteq D^\beta$;
  2. for every $c \in NI$, $c^\alpha = c^\beta$;
  3. for every $C \in NC$, $C^\alpha \subseteq C^\beta$;
  4. for every $R \in NR$, $R^\alpha \subseteq R^\beta$. 
\( \mathcal{KALC}^\infty \): Kripke-style semantics for \( \mathcal{ALC} \)

\( \mathcal{KALC}^\infty \)-models

\( \mathcal{KALC}^\infty \)-model \( K = \langle P, \leq, \rho, \iota \rangle \)

1. \( (P, \leq) \) is a \( K \)-poset with root \( \rho \);
2. \( \iota(\alpha) = (D^\alpha, \cdot^\alpha) \) such that, for every \( \alpha, \beta \in P \) with \( \alpha \leq \beta \):
   - \( D^\alpha \subseteq D^\beta \);
   - for every \( c \in \text{NI} \), \( c^\alpha = c^\beta \);
   - for every \( C \in \text{NC} \), \( C^\alpha \subseteq C^\beta \);
   - for every \( R \in \text{NR} \), \( R^\alpha \subseteq R^\beta \).

Forcing relation (examples)

\[ \alpha \models t : A, \text{ where } A \in \text{NC} \text{ iff } t^\alpha \in A^\alpha \]
\[ \alpha \models t : C \rightarrow D \text{ iff } \forall \beta \in P \text{ s.t. } \alpha \leq \beta, \beta \models \neg t : C \text{ or } \beta \models t : D \]
\[ \alpha \models t : \neg C \text{ iff } \forall \beta \in P \text{ s.t. } \alpha \leq \beta, \beta \models \neg t : C \]
\[ \alpha \models t : \forall R.C \text{ iff } \forall \beta \in P \text{ s.t. } \alpha \leq \beta \text{ and } \forall d \in D^\beta, \beta \models (t, d) : R \Rightarrow \beta \models d : C \]
**Kur and final states**

**Kur**

The condition on final states can be characterized by Kur:

\[ \text{Kur} : \forall R. \neg \neg A \rightarrow \neg \neg \forall R. A \]

**Kur axiom schema**

Let \( K \) be any instance of the axiom schema Kur in \( \mathcal{L} \) and let \( K = \langle P, \leq, \rho, \iota \rangle \) be a \( \mathcal{K}\mathcal{A}\mathcal{L}\mathcal{C}^\infty \)-model for \( \mathcal{L} \), then \( \rho \models K \).

**Final worlds \( \phi \)**

Maximal elements w.r.t. \( \leq \) relation

- \( \phi \) represents states of **complete knowledge**
- \( \phi \) behave like classical models: \( \phi \models c : A \sqcup \neg A \)
  
  this directly comes from forcing of \( \neg A \)!
Tableau calculus $\mathcal{T}$ for $\mathcal{KALC}^\infty$

**Tableau calculus $\mathcal{T}$**

- Refutation calculus: the negation of a formula cannot be satisfied
- Decompose sets of signed formulas $(T, F, F_c)$ building a proof tree

**Signed formulas**

- $T(A)$: $A$ is true;
- $F(A)$: $A$ is false;
- $F_c(A)$: $A$ will never be true;

S is contradictory iff $\{T(A), F(A)\} \subseteq S$, $\{T(A), F_c(A)\} \subseteq S$

**Features**

- Sound and complete w.r.t. $\mathcal{KALC}^\infty$ semantics
- $\mathcal{KALC}^\infty$ Disjunction Property
Relevant rules of $\mathcal{T}$

\[
\begin{align*}
S, T(t : A \sqcup B) & \quad \quad S, F(t : A \sqcup B) & \quad \quad S, F_c(t : A \sqcup B) \\
\hline
S, T(t : A) & \quad \quad S, T(t : B) & \quad \quad S, F(t : A) & \quad \quad S, F(t : B) & \quad \quad S, F_c(t : A) & \quad \quad S, F_c(t : B)
\end{align*}
\]
Relevant rules of $\mathcal{T}$

\[
\begin{align*}
S, T(t : A \sqcup B) & \quad S, T(t : A) \mid S, T(t : B) \\
\hline
S, F(t : A \sqcup B) & \quad S, F(t : A), F(t : B) \\
S, F_c(t : A \sqcup B) & \quad S, F_c(t : A), F_c(t : B)
\end{align*}
\]

\[
\begin{align*}
S, T(t : (A \sqcup B) \rightarrow C) & \quad S, T(t : A \rightarrow C), T(t : B \rightarrow C) \\
\hline
S, F(t : A \rightarrow B) & \quad S_p, T(t : A), F_c(t : B)
\end{align*}
\]
Relevant rules of $\mathcal{T}$

$$\frac{S, T(t : A \sqcup B)}{S, T(t : A) \mid S, T(t : B)}^{T\sqcup} \quad \frac{S, F(t : A \sqcup B)}{S, F(t : A), F(t : B)}^{F\sqcup} \quad \frac{S, F_c(t : A \sqcup B)}{S, F_c(t : A), F_c(t : B)}^{F_c\sqcup}$$

$$\quad \frac{S, T(t : (A \sqcup B) \rightarrow C)}{S, T(t : A \rightarrow C), T(t : B \rightarrow C)}^{T\rightarrow\Box} \quad \ldots \quad \frac{S, F_c(t : A \rightarrow B)}{S_p, T(t : A), F_c(t : B)}^{F_c\rightarrow}$$

$$\quad \frac{S, T(t : \forall R.A), T((t,s) : R)}{S, T((t,s) : R), T(s : A), T(t : \forall R.A)}^{T\forall}$$

$$\quad \frac{S, F(t : \forall R.A)}{S_p, T((t,p) : R), F(p : A)}^{F\forall^*} \quad \frac{S, F_c(t : \forall R.A)}{S_p, T((t,p) : R), F_c(p : A)}^{F_c\forall^*}$$
Properties of $\mathcal{T}$

Properties

It can be used to check **classic problems** on DLs: Concept validity, Subsumption, Instance checking
### Properties of $\mathcal{T}$

**Properties**

It can be used to check classic problems on DLs: Concept validity, Subsumption, Instance checking

**Disjunction property**

If $A \sqcup B \in \mathcal{KALC}^\infty$, then either $A \in \mathcal{KALC}^\infty$ or $B \in \mathcal{KALC}^\infty$

**Kur proof**

We can exhibit a validity proof for generic instances of Kur

$\rightarrow$ **Essential rule $F_c \forall$**: directly corresponds to condition on final states
About Finite Model Property

- if we omit condition on states, we obtain Int. Kripke semantics for $\mathcal{ALC}$ [de Paiva, 2005, Odintsov and Wansing, 2003]
- this version has an infinite countermodel for $\text{Kur} \Rightarrow$ no FMP! since $\text{Kur}$ is still valid in every finite model
- Still, we have no proof of FMP in $\mathcal{KALC}\infty$...
if we omit condition on states, we obtain Int. Kripke semantics for \( \mathcal{ALC} \) [de Paiva, 2005, Odintsov and Wansing, 2003]

- this version has an infinite countermodel for \( \text{Kur} \) \( \rightarrow \) no FMP!
  since \( \text{Kur} \) is still valid in every finite model

Still, we have no proof of FMP in \( \mathcal{KALC}^\infty \) ...

\[ \rightarrow \text{Search for a different Kripke semantics allowing a decidable tableaux procedure...} \]
Outline

1. Introduction
2. $BCDL$: information terms semantics for $ALC$
3. $KALC^\infty$: Kripke-style semantics for $ALC$
4. $KALC$: decidable Kripke-style interpretation for $ALC$
5. Application directions
**KALC**: decidable Kripke-style interpretation for \( \mathcal{ALC} \)

**KALC** [Bozzato et al., 2010, Bozzato, 2011]

- Kripke-style semantics for \( \mathcal{ALC} \)
- Tableaux calculus \( \mathcal{T}_K \) and proof search algorithm

**Features:**
- Kripke semantics for dynamic and incomplete knowledge
- Constructive reading of reasoning problems
- Reasoning problems solvable by proof search algorithm
**$\mathcal{KALC}$** Kripke semantics

**$\mathcal{KALC}$-model** $\mathcal{K} = \langle P, \leq, \rho, \iota \rangle$

- $(P, \leq)$ is a **finite** poset with root $\rho$;
- $\iota(\alpha) = (D^\alpha, \cdot^\alpha)$ such that, for every $\alpha, \beta \in P$ with $\alpha \leq \beta$:
  1. $D^\alpha \subseteq D^\beta$;
  2. for every $c \in \text{NI}$, $c^\alpha = c^\beta$;
  3. for every $A \in \text{NC}$, $A^\alpha \subseteq A^\beta$;
  4. for every $R \in \text{NR}$, $R^\alpha \subseteq R^\beta$. 

---

Notes

Forcing relation definition does not change.

$\mathcal{K}$ is valid also in every $\mathcal{KALC}$-model.

Syntax:

- limitations on formula definition

(Also, we only consider acyclic TBoxes)
**KALC** Kripke semantics

**KALC-model** $\mathcal{K} = \langle P, \leq, \rho, \iota \rangle$

- $(P, \leq)$ is a **finite** poset with root $\rho$;
- $\iota(\alpha) = (D^\alpha, \cdot^\alpha)$ such that, for every $\alpha, \beta \in P$ with $\alpha \leq \beta$:
  1. $D^\alpha \subseteq D^\beta$;
  2. for every $c \in NI$, $c^\alpha = c^\beta$;
  3. for every $A \in NC$, $A^\alpha \subseteq A^\beta$;
  4. for every $R \in NR$, $R^\alpha \subseteq R^\beta$.

**Notes**

- **Forcing relation** definition does not change
- **Kur** is valid also in every $\mathcal{K}$-$\cal{ALC}$-model
- **Syntax**: limitations on formula definition
  $$K ::= (s, t) : R \mid t : C \mid C \sqsubseteq D$$
  (Also, we only consider acyclic TBoxes)
Auditing example

Idea
- Kripke semantics can represent **partial** and **evolving** knowledge
- Example application domain: **auditing**

Auditing example [Mendler and Scheele, 2010]

**Problem domain:**
There are some companies, some of which may be insolvent

**An inference problem:**
Is a given company credit worthy? (Instance checking)
Auditing example: classical reasoning

Let us consider the following ABox \( A \):

\[
\begin{align*}
  & a : \text{Company} & b : \text{Company} & c : \text{Company} & d : \text{Company} \\
  & (a,b) : \text{hasCustomer} & (a,c) : \text{hasCustomer} \\
  & (b,c) : \text{hasCustomer} & (c,d) : \text{hasCustomer} \\
  & b : \text{Insolvent} & d : \neg \text{Insolvent}
\end{align*}
\]

A generic model \( M \) of \( A \) has the form:

\[
\begin{array}{ccc}
  \text{I} & \quad & \neg \text{I} \\
  \text{b} & \rightarrow & \text{c} & \rightarrow & \text{d} \\
  \text{a} & \rightarrow & \text{c} & \rightarrow & \text{d}
\end{array}
\]

\[
\begin{array}{cc}
  \text{I} & : \text{Insolvent} \\
  \text{hasCustomer} & : \text{hasCustomer}
\end{array}
\]
A company $x$ is **cw** *(Credit Worthy)* if:

$$I_x z \neg I_{cw y}$$

This clause corresponds to the formula

$$x : D \rightarrow cwD \equiv \exists hasCustomer.(Insolvent \sqcap \exists hasCustomer.\neg Insolvent)$$
Auditing example: classical reasoning

A company $x$ is $\text{cw}$ (Credit Worthy) if:

- $x$ has an insolvent customer $y$

This clause corresponds to the formula

$$x : \neg \text{ICW}$$

$L. \text{Bozzato} \ (\text{DKM} - \text{FBK-Irst})$
A company $x$ is cw (Credit Worthy) if:

- $x$ has an insolvent customer $y$
- such that $y$ can rely on at least one non-insolvent (i.e., solvent) customer $z$.

This clause corresponds to the formula
\[ D \rightarrow CW \]
\[ D \equiv \exists\ hasCustomer . (\neg Insolvent \land \exists\ hasCustomer . \neg Insolvent) \]
Auditing example: classical reasoning

A company $x$ is $\text{cw}$ (Credit Worthy) if:

- $x$ has an insolvent customer $y$
  such that $y$ can rely on at least one non-insolvent (i.e., solvent) customer $z$.

This clause corresponds to the formula

$$x : D \rightarrow \text{cw}$$

$$D \equiv \exists \text{hasCustomer.}(\text{Insolvent} \cap \exists \text{hasCustomer.}\neg \text{Insolvent})$$
Auditing example: classical reasoning

Let us add to the ABox $\mathcal{A}$ the formula

$$a : D \rightarrow CW$$

asserting the above property for $a$
Auditing example: classical reasoning

Let us add to the ABox $\mathcal{A}$ the formula

$$a : D \rightarrow CW$$

asserting the above property for $a$

**Instance checking problem**

Is the company $a$ Credit Worthy?

$\mathcal{A} \models a : CW$ ?
Auditing example: classical reasoning

We show that in any model $M$ of $\mathcal{A}$ we can find $y$ and $z$ such that:

$$I y \neg I z$$
Auditing example: classical reasoning

We show that in any model $\mathcal{M}$ of $\mathcal{A}$ we can find $y$ and $z$ such that:

CASE 1: $c$ is insolvent
Auditing example: classical reasoning

We show that in any model $\mathcal{M}$ of $\mathcal{A}$ we can find $y$ and $z$ such that:

CASE 1: $c$ is insolvent

CASE 2: $c$ is non-insolvent
Auditing example: classical reasoning

We show that in any model $\mathcal{M}$ of $\mathcal{A}$ we can find $y$ and $z$ such that:

CASE 1: $c$ is insolvent

CASE 2: $c$ is non-insolvent

$\mathcal{A} \models a: \exists \text{hasCustomer} . (\text{Insolvent} \cap \exists \text{hasCustomer}. \neg \text{Insolvent})$
Auditing example: classical reasoning

We show that in any model $\mathcal{M}$ of $\mathcal{A}$ we can find $y$ and $z$ such that:

CASE 1: $c$ is insolvent

CASE 2: $c$ is non-insolvent

$\mathcal{A} \models a : \exists \text{hasCustomer}.(\text{Insolvent} \land \exists \text{hasCustomer}. \neg \text{Insolvent})$

- $\mathcal{A} \models a : D$
- $\mathcal{A} \models a : D \rightarrow \text{CW}$ (since it belongs to $\mathcal{A}$)
- We conclude that: $\mathcal{A} \models a : \text{CW}$
We know that $a$ is $\text{cw}$, but we do not know why $\rightarrow \text{ALC}$ does not support constructive reasoning.

$\text{ALC}$ models cannot represent partial knowledge as:

$c$ is neither $\text{Insolvent}$ nor $\neg \text{Insolvent}$.
Auditing example: classical reasoning

Note

- We know that $a$ is CW, but we do not know why.
  $\Rightarrow$ $ALC$ does not support constructive reasoning.

- $ALC$ models can not represent partial knowledge as:
  $c$ is neither Insolvent nor $\neg$Insolvent.

$\Rightarrow$ We can represent partial and evolving knowledge, by Kripke semantics of $KALC$. 

$L. Bozzato$ (DKM - FBK-Irst)
Auditing example: constructive reasoning

\[ a: \text{Company} \quad b: \text{Company} \quad c: \text{Company} \quad d: \text{Company} \]
\[ (a, b): \text{hasCustomer} \quad (a, c): \text{hasCustomer} \]
\[ (b, c): \text{hasCustomer} \quad (c, d): \text{hasCustomer} \]
\[ b: \text{Insolvent} \quad d: \neg \text{Insolvent} \quad a: D \rightarrow \text{CW} \]

Let us consider the $\mathcal{KALC}$-model $K$ of $\mathcal{A}$:
The final worlds $\phi_1$ and $\phi_2$ are two states of complete knowledge:

$\phi_1 \models \neg c : \neg \text{Insolvent}$  
$\phi_2 \models c : \text{Insolvent}$

Instead, in $\rho$ we have partial knowledge about $c$:

$\rho \not\models c : \text{Insolvent}$  
$\rho \not\models \neg c : \neg \text{Insolvent}$
Auditing example: constructive reasoning

- We have that:

\[ \rho \models c: \text{Insolvent} \quad \square \neg \text{Insolvent} \]

and thus:

\[ \mathcal{A} \not\models c: \text{Insolvent} \quad \square \neg \text{Insolvent} \]

- In $\mathcal{KALC}$ we are not allowed to prove that $a$ is $\text{CW}$

From $\mathcal{A}$, we can not find insolvent $y$ and solvent $z$ in the definition of $\text{CW}$. 
Tableau calculus $\mathcal{T}_K$

To decide $\kaldc$-consequence relation $\models^k$ we introduce the tableau calculus $\mathcal{T}_K$

Tableau calculus $\mathcal{T}_K$

- Refutation calculus on signed formulas
- Signed formulas $T, F, T_s$ with:
  - $T_s(H)$: $H$ will be true from next states
- Realizability relation $K, \alpha \triangleright W$: evaluation of signed formulas in $\alpha$
  - $K, \alpha \triangleright T_s(H)$ iff $\alpha < \beta$ implies $\beta \models \neg H$

Theorem

Let $\mathcal{F}$ be a set of formulas and $q \in \mathbb{NI}$ not occurring in $\mathcal{F}$.

- $\mathcal{F} \models^k c : C$ iff $T(\mathcal{F}) \cup \{ F(c : C) \}$ is not realizable.
- $\mathcal{F} \models H$ iff $T(\mathcal{F}) \cup \{ T_s(c : \bot), F(H) \}$ is not realizable.
Tableau calculus $\mathcal{T}_\mathcal{K}$: relevant rules

- $\Delta, T(c : C \sqcup D) \quad T\sqcup$
  \[ \Delta, T(c : C) \mid \Delta, T(c : D) \]

- $\Delta, F(c : C \sqcup D) \quad F\sqcup$
  \[ \Delta, F(c : C), F(c : D) \]

- $\Delta, F(c : C \rightarrow D) \quad F\rightarrow$
  \[ \Delta, T(c : C), F(c : D) \mid \Delta_s, T(c : C), F(c : D) \]

- $\Delta, T(c : C \rightarrow D) \quad T\rightarrow$
  \[ \Delta, T(c : D) \mid \Delta, F(c : C), T_s(c : D) \mid \Delta_s, F(c : C), T_s(c : D) \]

- $\Delta, T(c : A), T(A \sqsubseteq C) \quad T\sqsubseteq$
  \[ \Delta, T(c : A), T(A \sqsubseteq C) \]

- $\Delta, T(c : \exists R.C) \quad T\exists$
  \[ \Delta, T((c, q) : R), T(q : C) \]

- $\Delta, T((c, d) : R), T(c : \forall R.C) \quad T\forall$
  \[ \Delta, T((c, d) : R), T(c : \forall R.C), T(d : C) \]

- $\Delta_s = \{ T(H) \mid T(H) \in \Delta \} \cup \{ T(H) \mid T_s(H) \in \Delta \}$
Tableau calculus $\mathcal{T}_K$: main theorem

Theorem (decidability)

Let $\Delta$ be a set of signed formulas:

- **Completeness**: $\Delta$ is realizable iff is not provable in $\mathcal{T}_K$.
- **Termination**: provability of $\Delta$ in $\mathcal{T}_K$ can be decided in finite time.
Tableau calculus $\mathcal{T}_K$: main theorem

**Theorem (decidability)**

Let $\Delta$ be a set of signed formulas:

- **Completeness**: $\Delta$ is realizable iff is not provable in $\mathcal{T}_K$.
- **Termination**: provability of $\Delta$ in $\mathcal{T}_K$ can be decided in finite time.

**Algorithm**

- **Input**: $\Delta$
- **Output**:
  - Closed proof table for $\Delta$
  - Or $\mathcal{KALC}$-model $K$ such that $K, \rho \models \Delta$

**Idea**:

- Build a countermodel for $\Delta$ by graph expansion
  (build a graph for each world of the model)
- If all attempts fail, we get a closed tableau for $\Delta$
Tableau calculus $\mathcal{T}_K$: algorithm

**REAL($\Delta$)**

\[ \mathcal{G}_\Delta = \langle N_\Delta, E_\Delta, PF_\Delta, SF_\Delta, TB, DF_\Delta \rangle; \]

if ( CONS($\mathcal{G}_\Delta$) == "inconsistent") then return "not realizable";
else return "realizable";

**CONS($\mathcal{G}$)**

while (exp-rules can be applied) and (PF clash-free) do begin
  pick $W \in (\mathcal{G})$;
  $\mathcal{G} = \text{exp-rules}(W, \mathcal{G})$;
end

if (PF contains a clash) then return "inconsistent";

while (succ-rules can be applied) do begin
  pick $W \in (\mathcal{G})$;
  $\mathcal{G}' = \text{succ-rules}(W, \mathcal{G})$;
  if (CONS($\mathcal{G}'$) == "inconsistent") then return "inconsistent";
end
return "consistent";
Tableau calculus $\mathcal{T}_K$: complexity

### $\text{PSPACE}$-hardness

- Straightforward translation between $\mathcal{KALC}$ and $\text{Int}$
- [Statman, 1979]: $\text{Int}$ decision problem is $\text{PSPACE}$-complete

$\Rightarrow$ $\mathcal{KALC}$ realizability is $\text{PSPACE}$-hard
Tableau calculus $\mathcal{T}_K$: complexity

**PSpace-hardness**
- Straightforward translation between $\mathcal{KALC}$ and $\text{Int}$
- [Statman, 1979]: $\text{Int}$ decision problem is $\text{PSPACE}$-complete
  $\rightarrow$ $\mathcal{KALC}$ realizability is $\text{PSPACE}$-hard

**PSpace-completeness?**
- Conjecture: for TBox $= \emptyset$, $\mathcal{KALC}$ is in $\text{PSPACE}$
**Tableau calculus $\mathcal{T}_K$: complexity**

**PSPACE-hardness**
- Straightforward translation between $\mathcal{KALC}$ and $\text{Int}$
- [Statman, 1979]: $\text{Int}$ decision problem is PSPACE-complete

$\Rightarrow \mathcal{KALC}$ realizability is PSPACE-hard

**PSPACE-completeness?**
- **Conjecture:** for TBox $= \emptyset$, $\mathcal{KALC}$ is in PSPACE
- **Idea:** trace technique
  - Countermodel can be exponential w.r.t. starting formulas...
  - ...but paths are independent and polynomial in size of input

$\Rightarrow$ Explore countermodel one path at a time requires space polynomial in size of input
Tableau calculus $\mathcal{T}_K$: complexity

**PSPACE-hardness**
- Straightforward translation between $\mathcal{KALC}$ and $\mathit{Int}$
- [Statman, 1979]: $\mathit{Int}$ decision problem is PSPACE-complete
  $\Rightarrow \mathcal{KALC}$ realizability is PSPACE-hard

**PSPACE-completeness?**
- Conjecture: for TBox $= \emptyset$, $\mathcal{KALC}$ is in PSPACE
- Idea: trace technique
  - Countermodel can be exponential w.r.t. starting formulas...
  - ...but paths are independent and polynomial in size of input
  $\Rightarrow$ Explore countermodel one path at a time requires space polynomial in size of input
- Still no formal proof for completeness of this strategy!
**KALC relations with KALC**

- **KALC** models are KALC models with a **possibly infinite poset**
- **Relations:**
  - \( KALC^\infty \subseteq KALC \), since every KALC model is a \( KALC^\infty \) model
  - \( KALC \subseteq KALC^\infty ? \)
    every \( KALC^\infty \) model has a corresponding KALC model?
Outline

1. Introduction
2. $BCDL$: information terms semantics for $ALC$
3. $KALC^\infty$: Kripke-style semantics for $ALC$
4. $KALC$: decidable Kripke-style interpretation for $ALC$
5. Application directions
Application directions

Starting points:

- **$BCDL$**: information terms semantics

- **$KALC$**: tableau procedure and Kripke semantics
Application directions

Starting points:

- **BCDL**: information terms semantics

- **KALC**: tableau procedure and Kripke semantics
Application directions

Starting points:

- **BCDL**: information terms semantics
  - Computational interpretation of proofs
  - Classical reading of descriptions
  - Natural notion of state
- **KALC**: tableau procedure and Kripke semantics

L. Bozzato (DKM - FBK-Irst)
Application directions

Starting points:

- **BCDL**: information terms semantics
  - Computational interpretation of proofs
  - Classical reading of descriptions
  - Natural notion of state

- **KALC**: tableau procedure and Kripke semantics
  - Comparison to tableau procedures for classical DLs
  - Basis for implementation
  - Representation of partial and dynamic knowledge
Application directions

Starting points:

- **BCDL**: information terms semantics
  - Computational interpretation of proofs
  - Classical reading of descriptions
  - Natural notion of state

- **KALC**: tableau procedure and Kripke semantics
  - Comparison to tableau procedures for classical DLs
  - Basis for implementation
  - Representation of partial and dynamic knowledge

Some current directions

- State description and action formalisms
- Semantic web service compositions and synthesis
App. directions: IT and states

IT and states

- Information terms encode a natural notion of state
- Used in [Ferrari et al., 2008] to represent system snapshots

→ Action formalism for $\mathcal{ALC}$ [Bozzato et al., 2009b]

An action formalism based on IT semantics of $\mathcal{BCDL}$
App. directions: IT and states

IT and states
- Information terms encode a natural notion of state
- Used in [Ferrari et al., 2008] to represent system snapshots

Action formalism for $\mathcal{ALC}$ [Bozzato et al., 2009b]
An action formalism based on IT semantics of $\mathcal{BCDL}$

System description and states
- **Theory $T$:** description of a system
  - **TBox:** system constraints (general properties)
  - **ABox:** current state of the system
- **State:** $\alpha \in IT(T)$
- **State consistency:** if there is a model $\mathcal{M}$ s.t. $\mathcal{M} \models \langle \alpha \rangle T$
App. directions: action language

- **Action:** \( P \Rightarrow Q \)

**Informal reading**
- If the **preconditions** \( P \) hold in a state \( \alpha \), the action can be applied.
- In the resulting state the **postconditions** \( Q \) must hold.

**Information content** \( IC(\langle \alpha \rangle T) \):
minimal set of atomic formulas encoding info. from \( \langle \alpha \rangle T \)
- **Applicability:** an action is **active** if \( P \subseteq IC(\langle \alpha \rangle T) \)
- **Action output** \( Out(\alpha) \): update \( IC(\langle \alpha \rangle T) \) with \( Q \)

**GENIT**
- Algorithm to build up a state (IT) for a system, given an action output
- It can be used to **trace reasons** for inconsistency
App. directions: web services composition

Service composition in \( BCDL \) [Bozzato and Ferrari, 2010]

- **Calculus** for definition of Semantic Web Services compositions
- Related to **program synthesis** in constr. logics [Miglioli et al., 1986]
- Services as combined functions "computing" information terms
Service composition in *BCDL* [Bozzato and Ferrari, 2010]

- **Calculus** for definition of Semantic Web Services compositions
- Related to *program synthesis* in constr. logics [Miglioli et al., 1986]
- Services as combined functions "computing" information terms

Composition calculus \( SC \)

\[
\begin{align*}
\Pi_1 : s_1(x) :: P_1 & \Rightarrow Q_1 \\
\vdots \\
\Pi_n : s_n(x) :: P_n & \Rightarrow Q_n
\end{align*}
\]

- **Applicability conditions (AC):** constraints for correctness of rule application
- **Computational interpretation (CI):** computational reading of logical rule
App. directions: web services composition

Service composition in $BCDL$ [Bozzato and Ferrari, 2010]

- Calculus for definition of Semantic Web Services compositions
- Related to program synthesis in constr. logics [Miglioli et al., 1986]
- Services as combined functions "computing" information terms

Composition calculus $SC$

\[
\begin{align*}
\text{s}(x) &: P \Rightarrow Q \\
\Pi_1 : \text{s}_1(x) &: P_1 \Rightarrow Q_1 \\
\vdots \\
\Pi_n : \text{s}_n(x) &: P_n \Rightarrow Q_n
\end{align*}
\]

- Applicability conditions (AC): constraints for correctness of rule application
- Computational interpretation (CI): computational reading of logical rule

Result

If a composition meets the ACs of its rules, then its computational interpretation is sound
Thank you for listening

Directions and Applications for Constructive Semantics in Description Logics

Loris Bozzato
bozzato@fbk.eu
https://dkm.fbk.eu/index.php/Loris_Bozzato

DKM - Data and Knowledge Management Research Unit,
FBK-Irst, Fondazione Bruno Kessler - Trento, Italy
*Kripke semantics and tableau procedures for constructive description logics.*

Composition of Semantic Web Services in a Constructive Description Logic.

A constructive semantics for $\mathcal{ALC}$.

A decidable constructive description logic.
To appear.

A note on constructive semantics for description logics.
Accepted at CILC09 - 24-esimo Convegno Italiano di Logica Computazionale.


References III

A compatible approach to temporal description logics.
In [Haarslev et al., 2010].

Paraconsistent description logics revisited.
In [Haarslev et al., 2010], pages 197–208.

Negations in description logic – contraries, contradictories, and subcontraries.

Towards a type system for semantic streams.

Towards Constructive DL for Abstraction and Refinement.


