OPPA European Social Fund
Prague & EU: We invest in your future.

Expressive Description Logics

Petr Křemen
petr.kremen@fel.cvut.cz

FEL ČVUT

Our plan

From $\mathcal{ALC}$ to OWL(2)-DL

Final Remarks

From $\mathcal{ALC}$ to OWL(2)-DL

Extending . . . $\mathcal{ALC}$ ...

- We have introduced $\mathcal{ALC}$, together with a decision procedure. Its expressiveness is higher than propositional calculus, still it is insufficient for many practical applications.
- Let's take a look, how to extend $\mathcal{ALC}$ while preserving decidability.

Extending . . . $\mathcal{ALC}$ ...

- We have introduced $\mathcal{ALC}$, together with a decision procedure. Its expressiveness is higher than propositional calculus, still it is insufficient for many practical applications.
- Let's take a look, how to extend $\mathcal{ALC}$ while preserving decidability.

Extending . . . $\mathcal{ALC}$ ... (2)

$\mathcal{N}$ (Number restructions) are used for restricting the number of successors in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\, R)$	$\left\{ a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\vert \geq n \right\}$
$(\leq\ n\, R)$	$\left\{ a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\vert \leq n \right\}$
$(=\ n\, R)$	$\left\{ a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\vert = n \right\}$

Example

- Concept $Woman \sqcap (\leq\ 3\ hasChild)$ denotes women who have at most 3 children.
- What denotes the axiom $Car \sqsubseteq (\geq\ 4\ hasWheel)$?
- ... and $Bicycle \equiv (=\ 2\ hasWheel)$?

Extending . . . $\mathcal{ALC}$ ... (2)

$\mathcal{N}$ (Number restructions) are used for restricting the number of successors in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\ R)$	$\left\{ a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\vert \geq n \right\}$
$(\leq\ n\ R)$	$\left\{ a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\vert \leq n \right\}$
$(=\ n\ R)$	$\left\{ a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\vert = n \right\}$

Example

- Concept $Woman \sqcap (\leq\ 3\ hasChild)$ denotes women who have at most 3 children.
- What denotes the axiom $Car \sqsubseteq (\geq\ 4\ hasWheel)$?
- ... and $Bicycle \equiv (=\ 2\ hasWheel)$?

Extending ... $\mathcal{ALC}$ ... (2)

$\mathcal{N}$ (Number restructions) are used for restricting the number of successors in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\, R)$	$\left\{ a \,\middle|\, \left\|\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\| \geq n \right\}$
$(\leq\ n\, R)$	$\left\{ a \,\middle|\, \left\|\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\| \leq n \right\}$
$(=\ n\, R)$	$\left\{ a \,\middle|\, \left\|\{b \mid (a,b) \in R^{\mathcal{I}}\}\right\| = n \right\}$

Example

- Concept $Woman \sqcap (\leq\ 3\ hasChild)$ denotes women who have at most 3 children.
- What denotes the axiom $Car \sqsubseteq (\geq\ 4\ hasWheel)$?
- ... and $Bicycle \equiv (=\ 2\ hasWheel)$?

Extending . . . $\mathcal{ALC}$ ... (3)

$\mathcal{Q}$ (Qualified number restrictions) are used for restricting the number of successors *of the given type* in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\, R\, C)$	$\left\{ a \,\middle|\, \left\| \{ b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}} \} \right\| \geq n \right\}$
$(\leq\ n\, R\, C)$	$\left\{ a \,\middle|\, \left\| \{ b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}} \} \right\| \leq n \right\}$
$(=\ n\, R\, C)$	$\left\{ a \,\middle|\, \left\| \{ b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}} \} \right\| = n \right\}$

Example

- Concept $Woman \sqcap (\geq\ 3\ hasChild\ Man)$ denotes women who have at least 3 sons.
- What denotes the axiom $Car \sqsubseteq (\geq\ 4\ hasPart\ Wheel)$?
- Which qualified number restrictions can be expressed in $\mathcal{ALC}$?

Extending … $\mathcal{ALC}$ … (3)

$\mathcal{Q}$ (Qualified number restrictions) are used for restricting the number of successors *of the given type* in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\, R\, C)$	$\left\{a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\vert \geq n\right\}$
$(\leq\ n\, R\, C)$	$\left\{a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\vert \leq n\right\}$
$(=\ n\, R\, C)$	$\left\{a \middle\vert\ \left\vert\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right\vert = n\right\}$

Example

- Concept $Woman \sqcap (\geq\ 3\ hasChild\ Man)$ denotes women who have at least 3 sons.
- What denotes the axiom $Car \sqsubseteq (\geq\ 4\ hasPart\ Wheel)$?
- Which qualified number restrictions can be expressed in $\mathcal{ALC}$?

Extending ... $\mathcal{ALC}$ ... (3)

$\mathcal{Q}$ (Qualified number restrictions) are used for restricting the number of successors *of the given type* in the given role for the given concept.

syntax (concept)	semantics
$(\geq\ n\ R\ C)$	$\left\{ a \middle| \quad \left|\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right| \geq n \right\}$
$(\leq\ n\ R\ C)$	$\left\{ a \middle| \quad \left|\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right| \leq n \right\}$
$(=\ n\ R\ C)$	$\left\{ a \middle| \quad \left|\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b^{\mathcal{I}} \in C^{\mathcal{I}}\}\right| = n \right\}$

Example

- Concept $Woman \sqcap (\geq\ 3\ hasChild\ Man)$ denotes women who have at least 3 sons.
- What denotes the axiom $Car \sqsubseteq (\geq\ 4\ hasPart\ Wheel)$?
- Which qualified number restrictions can be expressed in $\mathcal{ALC}$?

Extending . . . $\mathcal{ALC}$ ... (4)

$\mathcal{O}$ (Nominals) can be used for naming a concept elements explicitely.

syntax (concept)	semantics
$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$

Example

- Concept $\{MALE, FEMALE\}$ denotes a gender concept that must be interpreted with at most two elements. Why at most ?
- $Continent \equiv \{EUROPE, ASIA, AMERICA, AUSTRALIA, AFRICA, ANTARCTICA\}$?

Extending ... $\mathcal{ALC}$ ... (4)

$\mathcal{O}$ (Nominals) can be used for naming a concept elements explicitely.

syntax (concept)	semantics
$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$

Example

- Concept $\{MALE, FEMALE\}$ denotes a gender concept that must be interpreted with at most two elements. Why at most ?
- $Continent \equiv \{EUROPE, ASIA, AMERICA, AUSTRALIA, AFRICA, ANTARCTICA\}$?

laboratory Gerstner

$\ldots\mathcal{ALC}$, a co dál ? (5)

$\mathcal{I}$ (Inverse roles) are used for defining role inversion.

syntax (role)	semantics
R^-	$(R^{\mathcal{I}})^{-1}$

Example

- Role *maDite*$^-$ denotes the relationship *maRodice*.
- What denotes axiom $Person \sqsubseteq (= 2\ hasChild^-)$?
- What denotes axiom $Person \sqsubseteq \exists hasChild^- \cdot \exists hasChild \cdot \top$?

$\mathcal{I}$ (Inverse roles) are used for defining role inversion.

syntax (role)	semantics
R^-	$(R^\mathcal{I})^{-1}$

Example

- Role *maDite*$^-$ denotes the relationship *maRodice*.
- What denotes axiom $Person \sqsubseteq (= 2\ hasChild^-)$?
- What denotes axiom $Person \sqsubseteq \exists hasChild^- \cdot \exists hasChild \cdot \top$?

. . . $\mathcal{ALC}$, a co dál ? (5)

$\mathcal{I}$ (Inverse roles) are used for defining role inversion.

syntax (role)	semantics
R^-	$(R^{\mathcal{I}})^{-1}$

Example

- Role $maDite^-$ denotes the relationship $maRodice$.
- What denotes axiom $Person \sqsubseteq (= 2\ hasChild^-)$?
- What denotes axiom $Person \sqsubseteq \exists hasChild^- \cdot \exists hasChild \cdot \top$?

laboratory Gerstner

Extending . . . $\mathcal{ALC}$ ... (6)

$._{trans}$ (Role transitivity axiom) denotes that a role is transitive. Attention – it is not a transitive closure operator.

syntax (axiom)	semantics
$trans(R)$	$R^{\mathcal{I}}$ is transitive

Example

- Role *isPartOf* can be defined as transitive, while role *hasParent* is not. What about roles *hasPart*, $hasPart^{-}$, $hasGrandFather^{-}$?
- What is a transitive closure of a relationship ? What is the difference between a transitive closure of $hasDirectBoss^{\mathcal{I}}$ and $hasBoss^{\mathcal{I}}$.

laboratory Gerstner

Extending . . . $\mathcal{ALC}$ ... (6)

.*trans* (Role transitivity axiom) denotes that a role is transitive. Attention – it is not a transitive closure operator.

syntax (axiom)	semantics
$trans(R)$	$R^{\mathcal{I}}$ is transitive

Example

- Role *isPartOf* can be defined as transitive, while role *hasParent* is not. What about roles *hasPart*, $hasPart^{-}$, $hasGrandFather^{-}$?
- What is a transitive closure of a relationship ? What is the difference between a transitive closure of $hasDirectBoss^{\mathcal{I}}$ and $hasBoss^{\mathcal{I}}$.

$\mathcal{H}$ (Role hierarchy) serves for expressing role hierarchies (taxonomies) – similarly to concept hierarchies.

syntax (axiom)	semantics
$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

Example

- Role *hasMother* can be defined as a special case of the role *hasParent*.
- What is the difference between a concept hierarchy *Mother* $\sqsubseteq$ *Parent* and role hierarchy *hasMother* $\sqsubseteq$ *hasParent*.

Extending . . . $\mathcal{ALC}$ ...(7)

$\mathcal{H}$ (Role hierarchy) serves for expressing role hierarchies (taxonomies) – similarly to concept hierarchies.

syntax (axiom)	semantics
$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

Example

- Role *hasMother* can be defined as a special case of the role *hasParent*.
- What is the difference between a concept hierarchy $Mother \sqsubseteq Parent$ and role hierarchy $hasMother \sqsubseteq hasParent$.

Extending ... $\mathcal{ALC}$ ... (8)

$\mathcal{R}$ (role extensions) serve for defining expressive role constructs, like role chains, role disjunctions, etc.

syntax	semantics
$R \circ S \sqsubseteq P$	$R^{\mathcal{I}} \circ S^{\mathcal{I}} \sqsubseteq P^{\mathcal{I}}$
$Dis(R, R)$	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$
$\exists R \cdot Self$	$\{a \mid (a, a) \in R^{\mathcal{I}}\}$

Example

- How would you define the role *hasUncle* by means of *hasSibling* and *hasParent* ?
- how to express that R is transitive, using a role chain ?
- Whom does the following concept denote $Person \sqcap \exists likes \cdot Self$?

Extending . . . $\mathcal{ALC}$ ... (8)

$\mathcal{R}$ (role extensions) serve for defining expressive role constructs, like role chains, role disjunctions, etc.

syntax	semantics
$R \circ S \sqsubseteq P$	$R^{\mathcal{I}} \circ S^{\mathcal{I}} \sqsubseteq P^{\mathcal{I}}$
$Dis(R, R)$	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$
$\exists R \cdot Self$	$\{a \mid (a, a) \in R^{\mathcal{I}}\}$

Example

- How would you define the role *hasUncle* by means of *hasSibling* and *hasParent* ?
- how to express that R is transitive, using a role chain ?
- Whom does the following concept denote $Person \sqcap \exists likes \cdot Self$?

$\mathcal{R}$ (role extensions) serve for defining expressive role constructs, like role chains, role disjunctions, etc.

syntax	semantics
$R \circ S \sqsubseteq P$	$R^{\mathcal{I}} \circ S^{\mathcal{I}} \sqsubseteq P^{\mathcal{I}}$
$Dis(R, R)$	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$
$\exists R \cdot Self$	$\{a \vert (a, a) \in R^{\mathcal{I}}\}$

Example

- How would you define the role *hasUncle* by means of *hasSibling* and *hasParent* ?
- how to express that R is transitive, using a role chain ?
- Whom does the following concept denote $Person \sqcap \exists likes \cdot Self$?

Extending . . . $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

syntactic sugar	– axioms NegativeObjectPropertyAssertion, AllDisjoint, etc.
extralogical constructs	– imports, annotations
data types	– XSD datatypes are used

Extending . . . $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

    syntactic sugar – axioms NegativeObjectPropertyAssertion, AllDisjoint, etc.

    extralogical constructs – imports, annotations

    data types – XSD datatypes are used

Extending ... $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

    syntactic sugar – axioms NegativeObjectPropertyAssertion, AllDisjoint, etc.

    extralogical constructs – imports, annotations

    data types – XSD datatypes are used

Extending . . . $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

    syntactic sugar – axioms NegativeObjectPropertyAssertion, AllDisjoint, etc.

    extralogical constructs – imports, annotations

    data types – XSD datatypes are used

laboratory Gerstner

Extending . . . $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

    syntactic sugar – axioms NegativeObjectPropertyAssertion, AllDisjoint, etc.

    extralogical constructs – imports, annotations

    data types – XSD datatypes are used

laboratory Gerstner

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

    syntactic sugar – axioms NegativeObjectPropertyAssertion, AllDisjoint, etc.

    extralogical constructs – imports, annotations

    data types – XSD datatypes are used

Extending . . . $\mathcal{ALC}$ ... – OWL-DL a OWL2-DL

- From the previously introduced extensions, two prominent decidable supersets of $\mathcal{ALC}$ can be constructed:
  - $\mathcal{SHOIN}$ is a description logics that backs OWL-DL.
  - $\mathcal{SROIQ}$ is a description logics that backs OWL2-DL.
  - Both OWL-DL and OWL2-DL are semantic web languages – they extend the corresponding description logics by:

    syntactic sugar – axioms NegativeObjectPropertyAssertion, AllDisjoint, etc.

    extralogical constructs – imports, annotations

    data types – XSD datatypes are used

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs $(\mathcal{O}, \mathcal{N}, \mathcal{Q})$
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. $(\mathcal{H}, \mathcal{R}, \mathcal{I})$
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to
  - NEXPTIME for $\mathcal{SHOIN}$
  - N2EXPTIME for $\mathcal{SROIQ}$

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs $(\mathcal{O}, \mathcal{N}, \mathcal{Q})$
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in ∃-rule. $(\mathcal{H}, \mathcal{R}, \mathcal{I})$
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to
  - NEXPTIME for $\mathcal{SHOIN}$
  - N2EXPTIME for $\mathcal{SROIQ}$

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs ($\mathcal{O}, \mathcal{N}, \mathcal{Q}$)
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. ($\mathcal{H}, \mathcal{R}, \mathcal{I}$)
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to
  - NEXPTIME for $\mathcal{SHOIN}$
  - N2EXPTIME for $\mathcal{SROIQ}$

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs ($\mathcal{O}, \mathcal{N}, \mathcal{Q}$)
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. ($\mathcal{H}, \mathcal{R}, \mathcal{I}$)
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to
  - NEXPTIME for $\mathcal{SHOIN}$
  - N2EXPTIME for $\mathcal{SROIQ}$

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs ($\mathcal{O}, \mathcal{N}, \mathcal{Q}$)
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. ($\mathcal{H}, \mathcal{R}, \mathcal{I}$)
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to
  - NEXPTIME for $\mathcal{SHOIN}$
  - N2EXPTIME for $\mathcal{SROIQ}$

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs ($\mathcal{O}, \mathcal{N}, \mathcal{Q}$)
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. ($\mathcal{H}, \mathcal{R}, \mathcal{I}$)
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to
  - NEXPTIME for $\mathcal{SHOIN}$
  - N2EXPTIME for $\mathcal{SROIQ}$

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs $(\mathcal{O}, \mathcal{N}, \mathcal{Q})$
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in ∃-rule. $(\mathcal{H}, \mathcal{R}, \mathcal{I})$
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to
  - NEXPTIME for $\mathcal{SHOIN}$
  - N2EXPTIME for $\mathcal{SROIQ}$

Extending $\mathcal{ALC}$ – Reasoning

- What is the impact of the extensions to the automated reasoning procedure ? The introduced tableau algorithm for $\mathcal{ALC}$ has to be adjusted as follows:
  - additional inference rules reflecting the semantics of newly added constructs $(\mathcal{O}, \mathcal{N}, \mathcal{Q})$
  - definition of *R-neighbourhood* of a node in a completion graph. R-neighbourhood notion generalizes simple tests of two nodes being connected with an edge, e.g. in $\exists$-rule. $(\mathcal{H}, \mathcal{R}, \mathcal{I})$
  - new conditions for direct clash detection
  - more strict blocking conditions (blocking over graph structures).
- This results in significant computation blowup – from EXPTIME ($\mathcal{ALC}$) to
  - NEXPTIME for $\mathcal{SHOIN}$
  - N2EXPTIME for $\mathcal{SROIQ}$

Final Remarks

Other extensions

Modal Logic introduces *modal operators* – possibility/necessity, used in multiagent systems.

Example

- (□ represents e.g. the "believe" operator of an agent)

$$\Box(Man \sqsubseteq Person \sqcap \forall hasFather \cdot Man) \quad (1)$$

- As $\mathcal{ALC}$ is a syntactic variant to a multi-modální propositional logic, where each role represents the accessibility relations between worlds in Kripke structure, the previous example can be transformed to the modal logic as
-

$$\Box(Man \Rightarrow Person \wedge \Box_{hasFather} Man) \quad (2)$$

Vague Knowledge - fuzzy, probabilistic and possibilistic extensions (see [HPS05]).

Data Types ($\mathcal{D}$) allow integrating a data domain (numbers, strings), e.g. $Person \sqcap \exists hasAge \cdot 23$ represents the concept describing "23-years old persons".

Other extensions

Modal Logic introduces *modal operators* – possibility/necessity, used in multiagent systems.

Example

- ($\Box$ represents e.g. the "believe" operator of an agent)

$$\Box(Man \sqsubseteq Person \sqcap \forall hasFather \cdot Man) \quad (1)$$

- As $\mathcal{ALC}$ is a syntactic variant to a multi-modální propositional logic, where each role represents the accessibility relationa between worlds in Kripke structure, the previous example can be transformed to the modal logic as:
-

$$\Box(Man \Rightarrow Person \wedge \Box_{hasFather} Man) \quad (2)$$

Vague Knowledge - fuzzy, probabilistic and possibilistic extensions (see [HPS05]).

Data Types ($\mathcal{D}$) allow integrating a data domain (numbers, strings), e.g. $Person \sqcap \exists hasAge \cdot 23$ represents the concept describing "23-years old persons".

Other extensions

Modal Logic introduces *modal operators* – possibility/necessity, used in multiagent systems.

Example

- ($\Box$ represents e.g. the "believe" operator of an agent)

$$\Box(Man \sqsubseteq Person \sqcap \forall hasFather \cdot Man) \quad (1)$$

- As $\mathcal{ALC}$ is a syntactic variant to a multi-modální propositional logic, where each role represents the accessibility relationa between worlds in Kripke structure, the previous example can be transformed to the modal logic as:
-

$$\Box(Man \Rightarrow Person \wedge \Box_{hasFather} Man) \quad (2)$$

Vague Knowledge - fuzzy, probabilistic and possibilistic extensions (see [HPS05]).

Data Types ($\mathcal{D}$) allow integrating a data domain (numbers, strings), e.g. $Person \sqcap \exists hasAge \cdot 23$ represents the concept describing "23-years old persons".

Other extensions

Modal Logic introduces *modal operators* – possibility/necessity, used in multiagent systems.

Example

- ($\Box$ represents e.g. the "believe" operator of an agent)

$$\Box(Man \sqsubseteq Person \sqcap \forall hasFather \cdot Man) \quad (1)$$

- As $\mathcal{ALC}$ is a syntactic variant to a multi-modální propositional logic, where each role represents the accessibility relationa between worlds in Kripke structure, the previous example can be transformed to the modal logic as:
-

$$\Box(Man \Rightarrow Person \wedge \Box_{hasFather} Man) \quad (2)$$

Vague Knowledge - fuzzy, probabilistic and possibilistic extensions (see [HPS05]).

Data Types ($\mathcal{D}$) allow integrating a data domain (numbers, strings), e.g. $Person \sqcap \exists hasAge \cdot 23$ represents the concept describing "23-years old persons".

Other extensions

Modal Logic introduces *modal operators* – possibility/necessity, used in multiagent systems.

Example

- ($\Box$ represents e.g. the "believe" operator of an agent)

$$\Box(Man \sqsubseteq Person \sqcap \forall hasFather \cdot Man) \quad (1)$$

- As $\mathcal{ALC}$ is a syntactic variant to a multi-modální propositional logic, where each role represents the accessibility relationa between worlds in Kripke structure, the previous example can be transformed to the modal logic as:
-

$$\Box(Man \Rightarrow Person \wedge \Box_{hasFather} Man) \quad (2)$$

Vague Knowledge - fuzzy, probabilistic and possibilistic extensions (see [HPS05]).

Data Types ($\mathcal{D}$) allow integrating a data domain (numbers, strings), e.g. $Person \sqcap \exists hasAge \cdot 23$ represents the concept describing "23-years old persons".

DL Tools and Reasoners

RacerPro (`http://www.racer-systems.com`) is a commercial LISP-based system for OWL-DL and SWRL (also available in client/server version).

Pellet (`http://www.mindswap.org`) is an open-source Java OWL2-DL engine.

Jena `http://jena.sourceforge.net/` is an open-source Java framework and API for OWL and RDF(S).

FaCT++ `http://owl.man.ac.uk/factplusplus/` is a DL reasoner for $\mathcal{SHOIQ}$ written in C++.

and other ... KAON2, FOWL, Kris

DL Tools and Reasoners

RacerPro (http://www.racer-systems.com) is a commercial LISP-based system for OWL-DL and SWRL (also available in client/server version).

Pellet (http://www.mindswap.org) is an open-source Java OWL2-DL engine.

Jena http://jena.sourceforge.net/ is an open-source Java framework and API for OWL and RDF(S).

FaCT++ http://owl.man.ac.uk/factplusplus/ is a DL reasoner for $\mathcal{SHOIQ}$ written in C++.

and other … KAON2, FOWL, Kris

DL Tools and Reasoners

RacerPro (http://www.racer-systems.com) is a commercial LISP-based system for OWL-DL and SWRL (also available in client/server version).

Pellet (http://www.mindswap.org) is an open-source Java OWL2-DL engine.

Jena http://jena.sourceforge.net/ is an open-source Java framework and API for OWL and RDF(S).

FaCT++ http://owl.man.ac.uk/factplusplus/ is a DL reasoner for $\mathcal{SHOIQ}$ written in C++.

and other ... KAON2, FOWL, Kris

DL Tools and Reasoners

RacerPro (`http://www.racer-systems.com`) is a commercial LISP-based system for OWL-DL and SWRL (also available in client/server version).

Pellet (`http://www.mindswap.org`) is an open-source Java OWL2-DL engine.

Jena `http://jena.sourceforge.net/` is an open-source Java framework and API for OWL and RDF(S).

FaCT++ `http://owl.man.ac.uk/factplusplus/` is a DL reasoner for $\mathcal{SHOIQ}$ written in C++.

and other ... KAON2, FOWL, Kris

laboratory Gerstner

DL Tools and Reasoners

RacerPro (`http://www.racer-systems.com`) is a commercial LISP-based system for OWL-DL and SWRL (also available in client/server version).

Pellet (`http://www.mindswap.org`) is an open-source Java OWL2-DL engine.

Jena `http://jena.sourceforge.net/` is an open-source Java framework and API for OWL and RDF(S).

FaCT++ `http://owl.man.ac.uk/factplusplus/` is a DL reasoner for $\mathcal{SHOIQ}$ written in C++.

and other ... KAON2, FOWL, Kris

OPPA European Social Fund
Prague & EU: We invest in your future.
