Knowledge Interchange Format: Initial Motivations

• The Situation
  • The proliferation of intelligent systems
  • Each system with its own language
KIF: Initial Motivations

• The Need
  • To share and integrate information across diverse KR frameworks
**KIF: Initial Motivations**

- **The Problem**
  - General solution requires $n^2 - n$ translators.
  - The number of translators needed for integration thus grows exponentially with the development of new frameworks.

*Well, *one* problem anyway…*
KIF: Initial Motivations

• The Solution: KIF
  • A single “hub” framework, an universal *interlingua* “spoken” by every framework
  • Growth of translators to frameworks is linear ($t = 2n$)
dpANS KIF: The Revised Standard Version

- A standardized framework for expressing declarative information
  - Common syntax
  - Common, well-understood semantics
  - Full first-order expressiveness
    - Required in particular for metalinguistic constructs necessary to enable semantic integration
  - Powerful additional constructs
    - Variable Polyadicity
    - "Higher-order" syntax (predicate quantifiers)
    - Type-freedom (predicates double as terms)
    - Non-first-order expressiveness (sequence variables)
dpANS KIF: The Problems

- **The Uniformity Problem**
  - Both authoritarian and impractical to impose a single syntax on users

- **The Excess Baggage Problem**
  - Additional constructs often unneeded/undesired
  - Would be onerous to require them for conformance

- **The Formalization Problem**
  - No general model theory for dpANS KIF that accommodates all of its constructs
  - No proof theory, complete or otherwise
Approach to a Solution: Simplified Common Logic

- SCL is an abstract, generalized version of KIF
- SCL and the the problems of dpANS KIF
  - Uniformity
    - Standardize *structure* rather than any particular form
  - Excess Baggage
    - *Allow* rather than *impose* additional constructs
  - Formalization
    - SCL has a rigorous, model theory
    - SCL’s first-order component has a complete proof theory
    - SCL’s non-first-order component has an infinitary proof theory for sequence variables
      - SCL’s restrictions on sequence variables still permits limited automated reasoning
SCL: Lexicons

• An SCL lexicon shall include:
  • Nonlogical symbols
    • Predicate constants
      • A distinguished elements \( Id \)
    • Individual constants
    • Function symbols (\( FnSym \))
  • Variables
    • General variables
    • Sequence variables (possibly empty)
  • An \textit{arity} function on predicate constants and function symbols
    • \( p \) is an \( n \)-place predicate constant if \( \text{arity}(p) = n \) (\( P\text{Con}_n \))
    • \( p \) is \textit{variably polyadic} otherwise (\( P\text{Con}_\omega \))
    • \( f \) is an \( n \)-place function symbol if \( \text{arity}(f) = n+1 \) (\( F\text{n}\text{Sym}_n \))
    • \( f \) is \textit{variably polyadic} otherwise (\( F\text{n}\text{Sym}_\omega \))
Features of SCL Lexicons

- Type freedom permitted (not required)
  - No assumption predicates, constants, and function symbols are pairwise disjoint
    - Overlap can be partial or complete
    - Predicate constants can serve as individual constants or function symbols

- Variable polyadicity allowed
  - $arity$ is partial

- Points in the lexical spectrum
  - *Unconstrained* lexicons (~ classic KIF)
    - $arity = \emptyset$
    - predicates = constants = function symbols
  - *Traditional first-order (TFO)* lexicons
    - $arity$ total
    - Predicates, constants, function symbols are pairwise disjoint
**SCL Grammar: Terms**

- Let $Trm$ be the closure of the constants and individual variables under a term-forming operator $App$:

$$App : \bigcup_{n \in \mathbb{N}} ((FnSym_n \cup FnSym_\omega) \times Trm^n) \rightarrow Trm$$

- Only *structure* specified
- Concrete manifestations
  - $(\text{age-diff john (father-of john)})$
  - $\text{age-diff(john, father-of(john))}$
  - `<term>
      <fnsym>age-diff</fnsym>
      <indcon>john</indcon>
      <term>
        <fnsym>father-of</fnsym>
        <indcon>john</indcon>
      </term>
    </term>`
SCL Grammar: Predicables

• An \( n \)-place \textit{predicable} is anything that can be predicated of \( n \) arguments
  • Includes \( n \)-place and variably polyadic predicate constants and possibly general variables.
  • Inclusion of general variables allows such superficially “higher-order”, type-free constructs as:
    \[
    \text{(forall (\?x \?y \?F)}
    \text{(impl (Symmetric \?F)}
    \text{(impl (\?F \?x \?y) (\?F \?y \?x))})
    \]
  • I.e.,
    \[
    \forall F \forall x \forall y (\text{Symmetric}(F) \rightarrow (Fxy \rightarrow Fyx))
    \]
SCL Grammar: Atomic Formulas

• Let $Pred_n$ be the set of $n$-place predicables
• We specify only the structure of atomic formulas
• A predication operator $\text{Holds}$ on a lexicon is a one-one function on $Pred_n \times Trm^n$
• The range of $\text{Holds}$ is the set of atomic formulas (relative to the given lexicon)
SCL Grammar: Formulas I

- We specify only the *structure* of complex formulas via a set of operations
  - Id, Neg, Conj, Disj, Cond, EQ, UQ
- The operations are one-one and their ranges are pairwise disjoint
- Known collectively as *formula generators*
SCL Grammar: Formulas II

- The set $Fla$ of formulas is the closure of the atomic formulas under the formula generators

  - **Id** : $Trm \times Trm \rightarrow Fla$
  - **Neg** : $Fla \rightarrow Fla$
  - **Conj** : $Fla^* \rightarrow Fla$
  - **Disj** : $Fla^* \rightarrow Fla$
  - **Cond** : $Fla \times Fla \rightarrow Fla$
  - **EQ** : $(GVar \cup (GVar \times (PCon_1 \cup PCon_\omega)))^* \times Fla \rightarrow Fla$
  - **UQ** : $(GVar \cup (GVar \times (PCon_1 \cup PCon_\omega)))^* \times Fla \rightarrow Fla$
Concrete Instances

• The abstract structure ...

\[ UQ(\nu_1, \text{Cond}(\text{Holds}(\pi_1, \nu_1), \text{EQ}(\nu_2, \text{Conj}(\text{Holds}(\pi_2, \nu_2), \text{Holds}(\pi_3, \nu_1, \nu_2)))))) \]

• ...has such concrete instances as:
  
  

    (forall (?x)
      (impl (Boy ?x)
        (exists (?y)
          (and (Girl ?y)
            (Kissed ?x ?y)))))

  

• \( \forall x(\text{Boy}(x) \rightarrow \exists y(\text{Girl}(y) \land \text{Kissed}(x, y))) \)

• \([\text{every}^*x][\text{If}(\text{Boy} ?x)[\text{Then}:[*y](\text{Girl} ?y)(\text{Kissed} ?x ?y)]]) \)
Interpretations

- Interpretations of SCL languages consist four items:
  - A domain $I$ of individuals
  - A domain $R$ of relations
  - An function $ext$ that assigns extensions to relations
  - A denotation function $V$ that assigns elements of $I$ to individual constants and elements of $R$ to predicate constants and function symbols
Truth

• \textbf{Holds}(P,t_1,\ldots,t_n)\text{ is true in an interpretation just in case }\langle V(t_1),\ldots,V(t_n) \rangle \in \text{ext}(V(P))

• This semantics allows predicates that are also individual constants to hold of themselves:
  • \textbf{Holds}(P,P)\text{ is true in an interpretation just in case }\langle V(P) \rangle \in \text{ext}(V(P))

• Remaining clauses work as expected
  • Extra fiddling required for sequence variables
SCL and Traditional FOL

- SCL (in its current form) leaves the logical properties of standard first-order sentences intact.
- Properties can change only for languages that allow overlap of predicate constants and individual constants.
  - “Horrocks sentences”
    - $(\forall x)(P x \leftrightarrow \neg Q x) \land (\forall x y) x = y$
    - Inconsistent of $P$ and $Q$ also serve as individual constants
    - Consistent (as in TFO languages) unless that assumption is made
Translating into FOL

- Full SCL languages without sequence variables can be thought of as notational variants of first-order theories.
  - Introduce predicate $\text{Holds}_n$ and function symbol $\text{App}_n$, for each $n$
  - Atomic sentences: $(p \ t_1 \ldots \ t_n)^* = (\text{Holds}_n \ p \ t_1 \ldots \ t_n)$
  - Function terms $(f \ t_1 \ldots \ t_n)^* = (\text{App}_n \ f \ t_1 \ldots \ t_n)$
  - $(\text{foo} \ (g \ a) \ b \ (g \ f \ (f \ a)))^* = (\text{Holds}_3 \ \text{foo} \ (\text{App}_1 \ g \ a) \ b \ (\text{App}_2 \ g \ f \ (\text{App}_1 \ f \ a)))$