tag intersections in LKit

There are various situations where we need to check for consistency between syntactic forms. For example we might want to check that the use of a Preposition is consistent with the Noun-Phrase is associates with. The preposition "*in*" for example can be used with time ("*in the afternoon*") or containment ("*in the pond*") whereas "*under*" cannot be used with time ("*under the afternoon*" is not valid).

There are different ways to handle this agreement when building grammars (& a couple of different approaches which can be used with LKit). Here we descibe the following approach...

1. include a new slot for some words in the lexicon (the examples below use a slot called "tags";
2. use the Lisp/utils form $* to find set-intersections of tags;
3. use glitches to signal "no-agreement";
4. use Lkit's *Lisp* directive to build new slots in the semantic component of LKit grammar rules.

The example below specifies a simple grammar which uses 2 syntactic categories (type1 & type2). The grammar accepts sentences *S* as 2 word strings where the first word is a type1 and the second is a type2. The language has 3 words described as follows (note the tag forms)...

```
(build-lexicon
'((alpha  type1  (tags a b c))
  (beta   type2  (tags c d e))
  (delta  type2  (tags d e))
  ))
```

The grammar has one rule which checks there is an overlap/intersection of tags between its two words (& signals a glitch if there is not overlap) and builds a new tag slot for the resulting category *S* which is also the intersection of the tags of its words.

```
(build-grammar
 '((s  (s -> type1 type2)
       (glitch no-agreement
               if not type1.tags $* type2.tags)
       (tags . (lisp ($* type1.tags type2.tags)))
       )
   ))
```

The grammar operates as follows...

```
cg-user(4): (parse 's '(alpha beta))

_____________
complete-edge 0 2 s s (alpha beta)      nil
s    s -> (type1 type2)
Syntax
(s (type1 alpha) (type2 beta))
Semantics
(s (tags c))
t


cg-user(6): (parse 's '(alpha delta))

_____________
complete-edge 0 2 s s (alpha delta)     nil
     Glitches: (no-agreement)
s    s -> (type1 type2)
Syntax
(s (type1 alpha) (type2 delta))
Semantics
(s (tags))
t
```

Note that you can also do this kind of checking explicitly. In the following grammar s2 forms are built from type2 words as long as they have tags which include *b* or *c*.

```
(build-grammar
 '((s2  (s2 -> type2)
      (glitch no-agreement
              if not type2.tags $* '(b c))
      (tags . (lisp ($* type2.tags '(b c))))
      )
   ))
```

This grammar operates as follows...

```
cg-user(5): (parse 's2 '(beta))

_____________
complete-edge 0 1 s2 s2 (beta)          nil
s2   s2 -> (type2)
Syntax
(s2 (type2 beta))
Semantics
(s2 (tags c))
t
cg-user(6): (parse 's2 '(delta))
```

continued...

```
_____________
complete-edge 0 1 s2 s2 (delta)         nil
     Glitches: (no-agreement)
s2   s2 -> (type2)
Syntax
(s2 (type2 delta))
Semantics
(s2 (tags))
t
```