BSO Editor Design Notes

Version 0.1
Marc Verhagen, December 2005

1. Consistency Checking

There are three kinds of consistency checking:

1. consistency of individual types and entries: no type or entry may refer to non-existing types,
2. consistency of type hierarchy: enforce correct inheritance of roles,
3. consistency of an entry with its type: an entry may not refer in its rolemapping to a role that does not exist on the type (this means that an entry can become inconsistent with its type when the type's roles are changed).

The old BSO enforced the first two, but it did not enforce entry-type consistency. This means that the current ontology could be full of inconsistencies. We need to decide whether we want to enforce consitency type 3. If we do, we need to decide what it actually means. Is an entry never allowed to refer to a type that does not occur on the tpye? Is an entry allowed to not express roles on the type? My first hunch would be to say 'yes' and 'yes'.

Each time a type or entry is changed, we need to ensure local consistence (type 1 above), making sure that new or changed local attributes are valid. But some changes, that is, changes to types, have non-local effects and may introduce inconsistency of other objects, inconsistency of the type system or inconsistencies between types and entries. A change to a type can have repercussions of four kinds of objects in the ontology:

1. subtypes of T (types that occur in the tree underneath T)
2. entries of T and of subtypes of T
3. types that refer to T in their qualia structure, argument roles or matching roles
4. entries that refer to T in their bound arguments or preposition mappings

The table below lays out what needs to be done when a type T is changed. The columns indicate the kinds of changes that can be made to a type and the rows indicate the four objects listed above.

	rename to T'	**remove**	**change supertype**	**change attributes**
subtypes	direct subtypes of T change name of supertype to T'	remove	change inherited attributes	change inherited attributes
entries in subtree	no change	remove or move to user-specified type	check consistency with new inherited attributes on the	check consistency with new local or inherited attributes of the type

			type	
referring types	change value of referring attribute from T to T'	change value of referring attribute to user-specified type	no change	no change
referring entries	change value of referring attribute from T to T'	change value of referring attribute to user-specified type	no change	no change

I think that the core Python database wrapper needs to implement most of the functionality needed to make the above work.

1.1. Consistency and Multiple Editors

As I see it, we have two choices:

1. Introduce a single-editor mode, where some editor, more equal than others, gets to change the type system while no one else is allowed to make changes. This is the simplest way and has my preference. It could be implemented by allowing some people to grab a lock, make a bunch of edits and then release the lock. While the lock is in place, other users would see some kind of read-only icon in the browser, and the AddEnry buttons etc won't work. The disadvantage is the inconvenience to the editors. At least all entry editing should remain totally multi-user.
2. A pure multi-user mode for all occasions. We need to be much more careful for all those cases where two editors change things that are interdependent. This was a total pain for the old smalltalk interface, but would actually be less involved with our indivdual tpye/entry based personal playpen devoid BSO browser. Still, I'm hesitant.

2. Database Structure

Now is the time to make changes to the overal layout of the database. It is unlikely to happen later on, when we are actually editing the ontology. Here's my suggestion.

BSO_TYPE

typeID	VARCHAR, KEY	a unique ID for the type, for example t891
typeName	VARCHAR	a unique name for the type
onType	VARCHAR	the ID of the supertype
andType	NULL or VARCHAR	the ID of a 2nd supertype (for complex types)
cpa	BOOL	is the type used in CPA?
comment	VARCHAR	blahblah

editor	VARCHAR	the editor who last touched this type
timestamp	TIMESTAMP	time of last edit

Using IDs is not strictly necessary since the type name is unique. but it does make changing type names easier.

The cpa field is there so that CPA editors can leave flags. Those flags might be set by BSO people rather than CPA people though.

The editor field refers to the person who last made a change to some part of the type, possibly to another table like BSO_TYPE_QUALIA. Similarly, the timestamp refers to the last edit made to the type as a wole, not necessary to a value in this table.

BSO_TYPE_QUALIA

typeID	VARCHAR, FOREIGN KEY	a unique ID for the type
name	VARCHAR	name of qualia role
value	VARCHAR, FOREIGN KEY	value of role, the ID of a type
inherited	BOOL	is role inherited?

PrimaryKey = (typeID,name)

All qualia go here and the distiction between the two kinds of qualia gets ditched.

Currently, the roles and qualia are compiled out throughout the tree. That is, if a type X has qualia Q with value V, then a subtype Y of X will also have qualia Q with value V, and these values are stored in the table with Y. We may consider changing this. It would reduce the size of the databse and obviate the need to create consistency after some type edits. On the other hand, each time a type is displayed all its supertypes would need to be consulted. In any case, the inherited field is not needed if only local roles would end up in this table.

BSO_TYPE_ARGS

typeID	VARCHAR, FOREIGN KEY	a unique ID for the type
name	VARCHAR	name of argument role
value	VARCHAR, FOREIGN KEY	value of role, the ID of a type
inherited	BOOL	is role inherited?

PrimaryKey = (typeID,name)

Similar to qualia table. Same comment on the inherited field is relevant here.

BSO_TYPE_MATCHINGROLES

typeID	VARCHAR, FOREIGN KEY
role1	ENUMERATION
role2	ENUMERATION
type	VARCHAR, FOREIGN KEY

BSO_ENTRY

entryID	VARCHAR, KEY	a unique ID for the entry, for example e1788
stem	VARCHAR	name of the entry
type	VARCHAR, FOREIGN KEY	type of the entry, the ID of a type
tag	VARCHAR	syntactic category
cpa	BOOL or NULL	is this a good or bad exemplar for the type?
comment	VARCHAR	blahblah
editor	VARCHAR	the editor who last touched this entry
timestamp	TIMESTAMP	time of last edit

Perhaps we should have a separate table for CPA flags. So we would be able to collect multiple flags for an entry in case they change over time.

BSO_ENTRY_PREPMAPS

entryID	VARCHAR, FOREIGN KEY
type1	VARCHAR, FOREIGN KEY
role	ENUMERATION
type2	VARCHAR, FOREIGN KEY

type1 and type2 should be renamed in slightly more descriptive names, but I forgot what those two actually do and we're not touching prepositions yet

BSO_ENTRY_BOUNDARGS

entryID	VARCHAR, FOREIGN KEY
localRole	ENUMERATION
remoteRole	ENUMERATION
remoteType	VARCHAR, FOREIGN KEY

BSO_ENTRY_FEATURES

entryID	VARCHAR, FOREIGN KEY

name	VARCHAR
value	VARCHAR

BSO_REFERENCE_INDEX

referred	VARCHAR, FOREIGN KEY
referent	VARCHAR, FOREIGN KEY
type	ENUMERATION

ENUMERATION = (type, subtype, qualia, argument, boundarg, prepositionmapping, matchingrole).

This table is there to give quick acces to all types and entries that refer to a type. It contains all subtypes, all entires belong to the type etc,

BSO_CHANGE_LIST

id	VARCHAR, KEY
date	TIMESTAMP
editor	VARCHAR
type	NULL or VARCHAR
entry	NULL or VARCHAR
?	
?	

A list with all changes made to the ontology and lexicon. The smalltalk version implemented this by adding a revision number to each entry and type. Changes would not overwrite an existing table row but simply add one. Using a changelist that can be used to replay changes may be simpler. I don't know MySQL well enough but I don't think it has the same functionality that Oracle has, namely one where all changes since the last backup are recorded are available for recovery.

3. Other

Catherine says there's something fishy with none values, but that it is sufficiently isolated inside a python wrapper.

A stable core of python accessors is the interface between databse and GUI. Some additions may be needed to this core.