Andreas van Cranenburgh 0440949
Structures for Semantics, assignment 2,
University of Amsterdam, March 2010

Exercise 1

We have:

- every: $\langle\langle e,t\rangle,t\rangle\ \lambda P\lambda Q\forall x[P(x)\rightarrow Q(x)]$
- woman: $\langle e,t\rangle$
- loves: $\langle e,\langle e,t\rangle\rangle$
- john, mary: e
- or: $\langle e,\langle e,\langle\langle e,t\rangle,t\rangle\rangle\rangle\ \lambda x\lambda y\lambda P.(P(x)\vee P(y))$

We derive the following readings:

1. $(every(woman))(\lambda x.(AR2(loves)(x)(or(john)(mary))))$ [1]
   $\Rightarrow \forall x[woman(x)\rightarrow(loves(x,mary)\vee loves(x,john))]$
2. $or(john)(mary)((AR1(loves))(every(woman)))$
   $\Rightarrow \forall x[woman(x)\rightarrow loves(x,mary)]\vee\forall x[woman(x)\rightarrow loves(x,john)]$

Exercise 2

Imagine two worlds which disagree as to whether counting should start at zero or at one (they have not yet reached the compromise of starting at 0.5). This yields a counter-model, exploiting the fact that constants are non-rigid, so that they can refer to different values in the meta-language (right hand side):

$I(R)(w_0)=\{a\}$
$I(a)(w_0)(w_0)=90$
$I(a)(w_0)(w_1)=90$
$I(p)(w_0)(w_0)=90$
$I(p)(w_0)(w_1)=91$
$I(90)(w_0)=90$

$I(R)(w_1)=\{a\}$
$I(a)(w_1)(w_0)=91$
$I(a)(w_1)(w_1)=91$
$I(p)(w_1)(w_0)=90$
$I(p)(w_1)(w_1)=91$
$I(90)(w_1)=91$

[1] Without this lambda I could not succeed in getting the order of arguments right. Alternatively a typeshifting rule for switching the order of arguments could be added, so as to avoid this recourse to lambdas.

Exercise 3

(a)

(4a) IL: $\forall x[buy(mary, x) \rightarrow want(mary, ^\wedge (inexpensive(x) \wedge coat(x)))]$

Ty2: $\forall x[buy_a(mary_a, x) \rightarrow want_a(mary_a, \lambda v.(inexpensive_v(x) \wedge coat_v(x)))]$

(4b) IL: $\exists x[coat(x) \wedge want(mary, ^\wedge (buy(mary, x) \wedge inexpensive(x))]$

Ty2: $\exists x[coat_a(x) \wedge want_a(mary_a, \lambda v.(buy_v(mary_v, x) \wedge inexpensive_v(x))]$

(b)

Representation (5b) can be translated into Ty2 in the following manner:

$\lambda P(W_a(\lambda a.\exists x(P(x) \wedge B_a(x)(m_a)))(m_a))(I_a)$

Alpha-conversion:

$\lambda P(W_a(\lambda v.\exists x(P(x) \wedge B_v(x)(m_v)))(m_a))(I_a)$

Beta-conversion:

$W_a(\lambda v.\exists x(I_a(x) \wedge B_v(x)(m_v)))(m_a)$

This last expression has an equivalent surface form to (5a), so it must be logically equivalent as well. QED.

Representation (5c) can be translated into Ty2 in the following manner:

$\lambda Q(W_a(\lambda a.(\exists x(Q(a)(x) \wedge B_a(x)(m_a))))(m_a))(\lambda a I_a)$

Alpha-conversion:

$\lambda Q(W_a(\lambda v.(\exists x(Q(v)(x) \wedge B_v(x)(m_v))))(m_a))(\lambda a I_a)$

Beta-conversion:

$W_a(\lambda v.(\exists x(I_v(x) \wedge B_v(x)(m_v))))(m_a)$

This last expression is not equivalent to (5a), because this expression has I_v instead of I_a. As a counter-model, imagine a coat which is inexpensive in the actual world; however, in some other world, which happens to be the one where Mary buys the coat, it has become fabulously expensive due to the discovery that it once belonged to Catherine the Great. QED.

Representation (5d) can be translated into Ty2 in the following manner:

$\exists Q(Q(a) = I_a \wedge W_a(\lambda a.(\exists x(Q_a(x) \wedge B_a(x)(m))))(m_a))$

Alpha-conversion:

$\exists Q(Q(a) = I_a \wedge W_a(\lambda v.(\exists x(Q_v(x) \wedge B_v(x)(m_v))))(m_a))$

This last expression is clearly not equivalent to (5a), because the predicate denoting inexpensiveness (I) is outside of the scope of the existential quantifier, hence it cannot express the meaning expressed by (5a). QED.

Since (5d) has already been ruled out as an appropriate representation of sentence (4c), we are left with (5ab) and (5c). The difference between these boils down to whether the coat should be inexpensive in the actual world (right now), or in the world where Mary buys it (in the future). Obviously inexpensiveness matters most when a purchase is made, so I consider (5c) to be most appropriate.

Exercise 4

We will derive the following readings:

1. $loves(john, wife(john)) \wedge loves(bill, wife(bill))$
2. $loves(john, wife(john)) \wedge loves(bill, wife(john))$

Note that 'wife' is analyzed as a function of entities to entities; this has the obvious consequence that any given person can only have one wife, which would appear to be reasonable.

The formalism that we shall use is one that lexicalises all non-syntactic ambiguities. After the grammar arrives at a parse tree, all possible assignments of word meanings are generated (ie., the cartesian product of the possible meanings of the words in the sentence). A combination of a parse tree and a sequence of word meanings is then evaluated by percolating expressions upwards by repeated function application, which is either applicative or retro-applicative depending on inferred types; in case the types are incompatible the parse tree is discarded (this represents an incompatible choice of word meanings). When the root node is reached it will contain a well-formed expression, being the interperation of the sentence in question.

The lexicon:
john: $john\ e$
loves: $\lambda xy.loves(x, y)\ \langle e, \langle e, t\rangle\rangle$
his: $\lambda FLx.L(x, F(x))\ \langle\langle e, e\rangle, \langle\langle e, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle\rangle$
$\lambda FLxy.L(y, F(x))\ \langle\langle e, e\rangle, \langle\langle e, \langle e, t\rangle\rangle, \langle e, \langle e, t\rangle\rangle\rangle\rangle$
wife: $\lambda x.wife(x)\ \langle e, e\rangle$
and: $\lambda Pxy.(P(x)\&P(y))\ \langle\langle e, t\rangle, \langle e, \langle e, t\rangle\rangle\rangle$
bill: $bill\ e$
does_too: $\lambda xP.P(x)\ \langle e, \langle\langle e, t\rangle, t\rangle\rangle$

Note that for convenience 'does too' is analyzed as one word, which is justified by thinking of 'too' as a purely pragmatic operator, and we are not attending to pragmatics in this analysis.

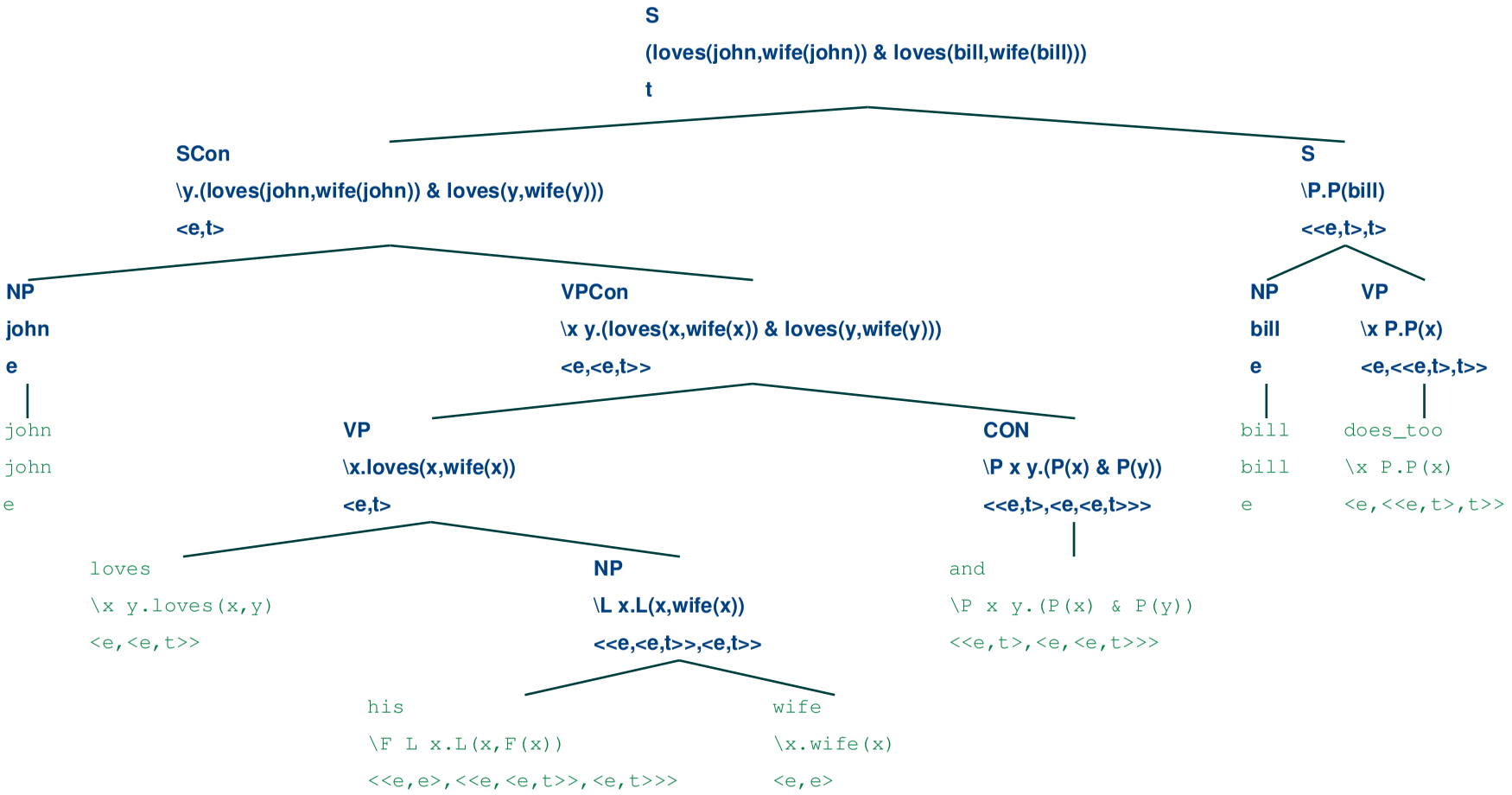
The types:
F, wife: $\langle e, e\rangle$,
P: $\langle e, t\rangle$,
L, loves: $\langle e, \langle e, t\rangle\rangle$

The grammar:

```
S -> NP VP | SCon S
SCon -> NP VPCon
PP -> 'with' NP
NP -> 'every' 'girl' | 'a' 'book' | 'a' 'boy' | 'his' 'wife' | NP PP
NP -> 'mary' | 'john' | 'bill'
VPCon -> 'smokes' CON | VP CON
VP -> 'hits' NP | 'loves' NP | 'does_too' | VP PP | DVP NP | NP VP
DVP -> 'gives' NP
CON -> 'and'
```

The following pages contain the syntax trees. Notice that the second reading requires a second application of 'John,' once as the referent for 'his,' and a second time as the subject of 'loves.' The first 'application' (in quotes, because the argument is not actually consumed[2]) is represented by a dashed line, since it is not part of the syntax proper, but an application triggered by the anaphor 'his,' through some psychological process of anaphor resolution not modelled here. The resulting tree is akward, but so is the reading it represents. In my opinion the strict reading is vanishingly marginal, and it would rather be elicited by an unproblematic sentence such as: "John loves his wife and Bill loves her too"

[2]Since the referent is not consumed as in normal function application, perhaps we should introduce a variaton on the lambda operator for this. Let's call this operator kappa. The second meaning of 'his' would then read:
his: $\lambda FL\kappa x\lambda y.L(y, F(x))\ \langle\langle e, e\rangle, \langle\langle e, \langle e, t\rangle\rangle, \langle e, \langle e, t\rangle\rangle\rangle\rangle$


S
(loves(john,wife(john)) & loves(bill,wife(bill)))
t
SCon
\y.(loves(john,wife(john)) & loves(y,wife(y)))
<e,t>
S
\P.P(bill)
<<e,t>,t>
NP
john
e
VPCon
\x y.(loves(x,wife(x)) & loves(y,wife(y)))
<e,<e,t>>
NP
bill
e
VP
\x P.P(x)
<e,<<e,t>,t>>
john
john
e
VP
\x.loves(x,wife(x))
<e,t>
CON
\P x y.(P(x) & P(y))
<<e,t>,<e,<e,t>>>
bill
bill
e
does_too
\x P.P(x)
<e,<<e,t>,t>>
loves
\x y.loves(x,y)
<e,<e,t>>
NP
\L x.L(x,wife(x))
<<e,<e,t>>,<e,t>>
and
\P x y.(P(x) & P(y))
<<e,t>,<e,<e,t>>>
his
\F L x.L(x,F(x))
<<e,e>,<<e,<e,t>>,<e,t>>>
wife
\x.wife(x)
<e,e>

S
(loves(john,wife(john)) & loves(bill,wife(john)))
t
SCon
\y.(loves(john,wife(john)) & loves(y,wife(john)))
<e,t>
S
\P.P(bill)
<<e,t>,t>
NP
bill
e
bill
VP
\x P.P(x)
<e,<e,t>,t>>
does_too
\x P. P(x)
<e, <<e, t >, t >>
VPCon
\x y.(loves(x,wife(john)) & loves(y,wife(john)))
<e,<e,t>>
CON
\P x y.(P(x) & P(y))
<<e,t>,<e,<e,t>>>
and
\P x y. (P(x) & P(y))
<<e, t >, <e, <e, t >>>
VP
\y.loves(y,wife(john))
<e,t>
NP
john
e
john
VP
\x y.loves(y,wife(x))
<e,<e,t>>
loves
\x y. loves(x, y)
<e, <e, t >>
NP
\L x y.L(y,wife(x))
<<e,<e,t>>,<e,<e,t>>>
his
\F L x y. L(y, F(x))
<<e, e>, <<e, <e, t >>, <e, <e, t >>>>
wife
\x. wife(x)
<e, e>

Exercise 5

conservatity	extension	quantity	example
yes	yes	yes	every
no	no	no	$(E - A) \subset (B \cap Prophets)$, ie., less non-A than B prophets
yes	no	no	$\lvert(E - A) \cap (B \cap Prophets)\rvert = 3$, ie., exactly three non-As are B prophets
no	yes	no	$A = (B \cap Prophets)$, ie., all and only A are B prophets
no	no	yes	$(E - A) = B$, ie., all and only non-As are B
yes	yes	no	$(A \cap John) \subset B$, ie., all John's A are B
yes	no	yes	$A \cup B = E$, ie., everyone is A or B
no	yes	yes	$A = B$, ie., all and only As are Bs

Proofs for the first two examples:

- Conservativity of "every"; this determiner passes the canonical conservativity test:

  every man smiles $\Leftrightarrow$ every man is a man that smiles

- Extension of "every": if we imagine two domains (eg., corresponding to a context of utterance), one a subset of the other, while the set of men and that of smiling people remains the same, then "every man smiles" is equivalent in both domains, which is the condition for extension.

- Quantity of "every": permuting elements of the domain, if done systematically, does not change the meaning of "every", because it depends only on the elements of A and B, not on any other non-permuted predicate, hence it has quantity.

- Conservativity of "there are less non-As than B prophets"; this quantifier obviously fails the conservativity test:

  There are less non-men than smiling prophets $\not\Leftrightarrow$ There are less non-men than non-men that are smiling prophets

- Extension of "there are less non-As than B prophets": the definition clearly references the domain E by referring to "non-As", so it is likely to fail the extension test. To see this, imagine one domain with two smilers, one man and one woman; a second domain also has two smilers and one man, but two women; everyone is a prophet in both domains. Now, in the first domain the quantifier is true, yet in the second it is false, which proves that it is context-dependent.

- Quantity of "there are less non-A than B prophets": Imagine a *deus ex machina* upsetting the order of things by permuting the set of men and smiling people. Beforehand we have:

  I(men) = { Moses, Abraham }
  I(prophet) = { Abraham, Moses }
  I(smilers) = { Abraham, Sarah }
  I(women) = { Sarah }

  Clearly the quantifier is false before the permutation, because there is one non-man but also one smiling prophet. However, after permuting the domain of the smilers and the men (yet crucially not that of the prophets) the quantifier is true:

  I(men) = { Sarah, Abraham }
  I(prophet) = { Abraham, Moses }
  I(smilers) = { Abraham, Moses }
  I(women) = { Moses }

Exercise 6

a)

Anti-veridicality applies to questions, but I would expect it to also apply to the non-specific irrealis, so it is of no use here.

Monotonicity: *any* as an NPI is allowed in all downward entailing contexts, so it is a downward monotonic determiner, whereas most other determiners are upward monotonic. However, for *any* in the free choice sense this fact is of no use.

Specifity would appear to me to be the best candidate, were it somehow possible to combine it with the irrealis. Alternatively, the anti-additivity property combined with question contexts could describe the funtions of *any* parsimoniously. The latter option seems to me to be superior, because questions are also pragmatically very different from the rest of the contexts for which *any* is licensed.

b)

- #IR: #You must try anything

  Predicted meaning: $must(\forall x[only_x try(x)])$, ie., $\perp$ This sentence appears to use *any* as free choice. However, the exclusivity condition makes it a contradicion, because it is not possible to do everything and only one thing at once.

  By the way, what about the following sentence, which seems also to be an irrealis yet the use of *any* seems appropriate:

  I wish I had any money, yet I don't

- CA: If anything goes wrong, call the neighbors

  Predicted meaning: $\exists x[wrong(x) \wedge happens(x)] \Rightarrow [...]$ This meaning does not have any negation, neither covert nor overt, so our account does not permit it.

- IN: It is not necessary that anybody call.

  Predicted meaning: $\neg\exists x[must(call(x))]$. This meaning has *any* bound to a negation operator, so it is correctly predicted to be correct by our account.

- CO: John is taller than anyone else

  Predicted meaning: $\forall x[tallness(john) > tallness(x)]$ This is a universal usage of *any*, yet without the exclusivity operator. This implies it is incomptable with our account.

∞