CORRECTIONS TO THE TEXT

- p. vii: Section 5 of Chapter 1 should be called "Narrowing the Definition of Language Further".
- p. 5: The colon in the fifth line after the box should be in normal font, not in Courier New.
- p. 6: "find a lot about the known languages" should read "find out a lot about the known languages".
- p. 7, line –9: Change "Either by its designers or by its users" to "Either by their designers or by their users".
- p. 12, line 16: italicize "antecedent".
- p. 25: Section 5 should be called "Narrowing the Definition of Language Further".
- p. 26, line -3: "grew so quickly that" should read "grew so quickly with rank that".
- p. 27, footnote 7, line 3: italicize the exponent x.
- p. 28, line -6: Change "to measure of the difficulty" to "to measure the difficulty".
- p. 36, line 2: Decapitalize "Spanish" to "spanish".
- p. 36, line -1: change the first occurrence of "eve" to "eva" (the second one should remain as is).
- p. 41, the last paragraph of section 9 should read as follows: The grammars we have been developing can also deal comfortably with these issues. The reason for this is that grammatical statements may differ with respect to sound without having to differ with respect to meaning (and, at the same time, may differ with respect to meaning without having to differ with respect to sound). In other words, grammatical statements may specify sounds and meanings independently of each other (they are ***independent parameters*** of the language relation), thus providing ***modular*** accounts of each. Sample statements that may deal with synonymy and homonymy can be found in (23) and (24), respectively.
- p. 43, line –7: Italicize *grammar*.
- p. 48, line -2: "*niñas alto*" should read "*niñas, alto*"
- p. 51, line -10: Delete the second occurrence of "only".
- p. 60, line 17: It should say "Program 6" rather than "Program 5".
- p. 64: Delete the second occurrence of "`j`" in the parenthesis in **R2**. This is the one between the upside-down "r" and the "w".
- p. 66, l. 4: The colon should be followed by a dash (as all the other colons on this page).
- p. 69: The second sentence of footnote 10 should read: "If so, enter the following (at the prompt):"
- p. 73, Problem 2: add "(if any)" after "give the instantiation of the variables". Also change "instantiation" to "instantiations".
- p. 74, Problem 4: Replace the beginning of the prompt as follows: "Use your understanding of the `[Head|Tail]` representation of lists to explain why the following PROLOG procedure defines…"
- p. 74: After revising Problem 4 as indicated above, reorder problems 3 and 4 [and reflect this reordering in the Answer Key that is to accompany the textbook].

- p. 81, fn. 3: Invert the order of the sentences (the second one should be first and the first, second).
- p. 83, line -13: change "number.swipl" to "num.swipl"
- p. 86, line 2: change "111" to "97".
- p. 89, line -2: The line is not right-justified (it does not end flush right).
- p. 90, line -9: Change "derived from" to "broken down into".
- p. 101, line 13: Replace "shares" by "has".
- p. 115: The sentence under (71) should be split in two so as to read "According to the received view, (70) evaluates a situation in positive terms while (71) casts it in a negative light instead. *Yet, both are saying the same thing*."
- p. 116, line 10: Replace "mentioned in lecture" by "mentioned in the preceding text".
- p. 116, Problem 6, between "If so, which?" and "You may assume that…": Add "Can you argue for your decompositions by appealing to semantic intuitions like redundancy, contradiction, or entailment?"
- p. 118, line 15: Replace "Second, that both words and meanings" by "Second, that both words and phrases"
- p. 130, line 12: Change "thus program" to "this program".
- p. 132, line 20: An underscore is missing between the stroke and the right bracket. In other words, it should read "[person | _]" rather than "[person |]"
- p. 139, line -4: Place "append" in Courier New font (or in the "Typewriter-looking font" code is placed in).
- p. 146, line 15: Change "maybe" to "may be".
- p. 152, line -13: Replace "forms a noun out of a verb" by "forms a verb out of a noun".
- p. 152, line 9: Replace "devoicing" by "voicing".
- p. 167: There is a spurious line within (5).
- p. 173, line -3: Replace "acoustic parameters" by "articulatory parameters".
- p. 176, line -7: Add the following sentence to the paragraph ending in this line: To make matters more perspicuous, enter

  ```
  english([X,Y]), nas(Y).
  english([X,Y]), name(X, [_,771]).
  ```

  The first query asks what vowels X can precede a nasal Y; the second asks what consonants Y can follow a nasal vowel X.
- p. 182, lines 14 and 18: The second apostrophe of ‘X' should to be curved rather than straight.
- p. 182, line -5: Replace "T" by "Y'.
- p. 182, line -6: This line should end in a period.
- p. 184, lines 5-7: Replace "S=" by "E="
- p. 198, line 9: Replace "over sets of elements of X that are big relative to X" by "over subsets of those sets of things (namely those subsets that consist of things that are big as far as the things in those sets are concerned)."
- p. 199, line –2: Replace "singular/feminine" by "singular/plural").

- p. 207: The paragraph that begins "To gather the effect" fails to indent. Moreover, a line is needlessly skipped between this paragraph and the preceding one.
- p. 208: Ditto for the paragraph that begins "For starters, notice that".
- p. 219, line 4: Replace "(2)" by "(3)".
- p. 221: Center the root of the tree (relative to the tree).
- p. 222, l. 6: Change "an variable" to "an unnamed variable".
- p. 225, l. –13: Change "mean relative to a" to "refer relative to any".
- p. 233, line -13: Change "refer relative to any" to "refer to relative to any".
- p. 238: indent as usual lines two, three, and four of (3).
- p. 257, Exercise 2, line 2: Replace *y'all* by *yall*.
- p. 258, lines1 and 2: Replace *a walk on the park* by *a walk in the park*.
- p. 262, Footnote 3: Change *We will improve* to *You will have a chance to improve*.
- p. 297, Exercise 4, line 3: Change *are wont do* to *are wont to do*.
- p. 297, line –8: Change `"passive.swipl"` to `"voice.swipl"`
- pp. 301-302, Figures 1, 2, 4: Match their fonts to those of the rest of figures in this chapter.
- p. 337 (Appendix F): The chart is missing the last 6 columns (see attached file; these columns were cut when the page setup was shifted from landscape to portrait).
- p. 345: Add the following between lines 9 and 10 (and skip a line before and after this material):

```
spanish([adan], [nounphrase], [adam],
[entity]).
```

- p. 355: The first two statements of **PROGRAM 2** should be switched (so that the first four lines of code of this program should be):

```
 spanish([adan], [nounphrase], [adam],
[entity]).

spanish([eva], [nounphrase], [eve],
[entity]).
```

- p. 433, between MONTLEY and NIDA: Add the following reference:
  Moschovakis, Yiannis (1993) "Sense and denotation as algorithm and value".
  In Oikkonen, J. & Väänänen, J. (Eds.) Logic Colloquium 90. Berlin, Springer-Verlag. pp. 210-249.

ADDITIONS TO THE TEXT

[The material below should replace the third paragraph from the end of Section 6, Chapter 2. To make the passage easier to identify, the old and the new paragraph begin and end the same way; the addition is in the middle]

To count how many syllables are in this list, we may issue a complex query consisting of this `findall` query and a `length` query:

```
findall(Syllable, syllable(Syllable), List), length(List,Length).
```

Upon entering this query, PROLOG will again list all the syllables for us. But now it will go on to tell us how many there are. To have PROLOG just count the syllables without listing them we can enter the following query instead:

```
aggregate_all(count, syllable(_), Count).
```

Either way, PROLOG will respond by telling us that `syllable.swipl` can generate 20,608 syllables. Yet, it can efficiently accept well-formed syllables like [s, p, I, n] and reject ill-formed ones like [p, s, I, n]. To verify this, enter the following:

```
syllable([s, p, ɪ, n]).
syllable([p, s, ɪ, n]).
```

[The paragraph below should go in Chapter 3, Section 3. There it should go between the paragraph that discusses example (12) and the paragraph that begins "Fortunately, we do not"]

One interesting argument for the metalinguistic version of indirect referentialism comes from the fact that a sentence like *A Robert Jones is here to see you* does not mean that someone with the identity of Robert Jones is here to see you. Or even that one of several individuals with identities similar to him is here to see you. It only means that someone named *Robert Jones* is here to see you. Or that a bearer of the name *Robert Jones* is here to see you. But this is exactly what one would expect from the metalinguistic version of indirect referentialism (and the opposite of what one would expect from direct referentialism).

[The material below should appear as Section 12 of Chapter 3. This is an entirely new section. All subsequent sections of that chapter should of course be renumbered accordingly. And the Table of Contents (and its page numbers) should be updated to reflect all of these changes as well. References to all the affected sections should be changed as well. Fortunately, there is only one such reference. It is in the first footnote of Chapter 6, which should now mention Section 13 rather than 12]

12. The High Resolving Power of Procedural Semantics

So the procedural theory of meaning is grained even more finely than possible-world semantics. So much so, in fact, that it can draw semantic distinctions which have largely escaped notice in the literature. Consider for example the difference between English *grandfather* and Spanish *abuelo*. While the English kinterm conveys the notion of *parent father* (58), its Spanish counterpart conveys the notion of *male grandparent* instead (59).

```
(58)    parentfather(A,B):-               (59)   malegrandparent(A,B):-
             father(A,C),                             male(A),
             parent(C,B).                             grandparent(A,B).
```

These concepts would be incorporated in the respective referential queries of *grandfather* and *abuelo* as follows:

```
(60)    R :: findall([A,B], parentfather(A,    B), R).
(61)    R :: findall([A,B], malegrandparent(A, B), R).
```

And these queries—or rather the referential procedures they trigger—would be assigned as meanings to our two kinterms. Since the procedures triggered by (60) and (61) are different, our two kinterms would be assigned—sight unseen—different meanings. Notice that this is a welcome result, as the two sentences below have different cognitive values.

(62) *A father of a parent is a father of a parent.*
(63) *A father of a parent is a male grandparent.*

Interestingly, however, although *grandfather* and *abuelo* have different meanings, they will always have the same reference (and this might explain why their semantic difference has hitherto escaped notice).

To prove that *grandfather* and *abuelo* will always have the same reference, let us decompose them all the way down to the primes they involve. Since *father* decomposes into *male parent* and *grandparent* into *parent of a parent*, we have that

```
(58')   fatherparent(A,B):-               (59')  malegrandparent(A,B):-
             male(A),                                 male(A),
             parent(A,C),                             parent(A,C),
             parent(C,B).                             parent(C,B).
```

What these ***prime decompositions*** show is that the two relations hold if (and only if) the same three conditions hold in each case. And that the only diference between said relations is that (58) conjoins the first two of these conditions while (59) conjoins the last two. But since conjunction is associative, the two concepts will hold under exactly the same circumstances (i.e. in the same models).

We will say that two expressions are ***semantically equivalent*** if and only if they will have the same reference in every model. We will say that they are ***semantically identical*** (or ***synonymous***) if they are assigned the same referential query. *Grandfather* and *abuelo* would therefore be semantically equivalent without being semantically identical.

Similar situations arise with *grandmother* and Spanish *abuela*; with *grandson* and Spanish *nieto*; and with *granddaughter* and Spanish *nieta*. Beyond kinterms, semantic distinctions have also escaped notice for numerals. Take for instance English *eighty* and French *quatre-vingts*. They would convey the concepts `eighttens` and `fourtwenties` defined below.

(64)
```
eighttens(A):-
    A is 8 * 10.
```

(65)
```
fourtwenties(A):-
    A is 4 * 20.
```

And these concepts would be incorporated in the respective referential queries of *eighty* and *quatre-vingts* as follows:

(66) `R :: eighttens(R).`
(67) `R :: fourtwenties(R).`

So English *eighty* and French *quatre-vingts* would not be assigned the same meanings. Consequently, they would not be semantically identical. Yet, they would still be semantically equivalent, as the following equations are true in every model.

$$8 \cdot 10 = (4 \cdot 2) \cdot 10 = 4 \cdot (2 \cdot 10) = 4 \cdot 20$$

Or at least in every model in which standard arithmetic holds. Again, this is a welcome result, as (68) and (69) are not equally contentful.

(68) *Eight tens is four twenties.*
(69) *Eight tens is eight tens.*

One interesting consequence of the foregoing discussion is that translation is not always the replacement of an expression in one language by a semantically identical expression from another; sometimes it cannot be more than the substitution of one expression by a mere semantic equivalent.

Beyond this, the procedural theory of meaning is able to make subtle distinctions of connotation and emphasis, distinctions which are generally considered too fine for denotational semantics to make. Consider for example a familiar contrast in connotation:

(70) *The glass is half full.*
(71) *The glass is half empty.*

According to the received view, (70) evaluates a situation in positive terms, (71) casts it in a negative light instead, *but both are saying the same thing*. As procedural semantics would have it, however, the two sentences in question are not saying the same thing, as they are conveying different procedures for the computation of a truth value. Even if both procedures end up computing the same one. The reason for this is plain: one of these procedures involves the concept (or subprocedure) of fullness, while the other one involves the concept (resp. subprocedure) of emptiness. And these concepts better be different. Whether procedural semantics will be furthermore able to relate, as desired, fullness to a constellation of positives and emptiness to one of negatives is something that remains to be seen.

The reason procedural semantics was able to make the distinction between (70) and (71) is that a procedure is the sequence of steps followed in the computation of a value. To convey a procedure is therefore to convey these steps. All of them. This fact also explains why procedural semantics can account for differences of emphasis. Compare (72) and (73).

(72) *John said so.*
(73) *John himself said so.*

Although the interpretation of reflexives is complex, it is safe to assume that (72) involves a single reference to John, while (73) involves a double reference to him. If the number of references to John are retained (as they must in the procedures for the interpretations of these sentences), then we have a distinction between them. And pretty much the correct one: (73) is an emphatic version of (72) because (73) refers to John twice, while (72) does so only once.

[The material below should go right after the last paragraph of current Section 2, Chapter 4]

For a simpler example of a grammar fragment with referential queries as meanings, consider the following variation of `mandarin.swipl`, which we introduced in Chapter 1, Section 6.

```
:- ['doublecolon.swipl'].

mandarin([yi ], numeral, R :: R is 1, [number, unit, first]).
mandarin([er ], numeral, R :: R is 2, [number, unit, nonfirst]).
mandarin([san], numeral, R :: R is 3, [number, unit, nonfirst]).
mandarin([si ], numeral, R :: R is 4, [number, unit, nonfirst]).
mandarin([wu ], numeral, R :: R is 5, [number, unit, nonfirst]).
mandarin([liu], numeral, R :: R is 6, [number, unit, nonfirst]).
mandarin([qi ], numeral, R :: R is 7, [number, unit, nonfirst]).
mandarin([ba ], numeral, R :: R is 8, [number, unit, nonfirst]).
mandarin([jiu], numeral, R :: R is 9, [number, unit, nonfirst]).
mandarin([shi], numeral, R :: R is 10,[number, ten,  first]).

mandarin(A, numeral, R :: R is N1 * N2,[number, ten, nonfirst]):-
   mandarin(C, numeral, R1 :: R1 is N1,[number, unit,nonfirst]),
   mandarin(E, numeral, R2 :: R2 is N2,[number, ten, first]),
   append(C, E, A).

mandarin(A, numeral, R :: R is N1 + N2,[number]):-
   mandarin(C, numeral, R1 :: R1 is N1,[_,ten,_]),
   mandarin(E, numeral, R2 :: R2 is N2,[_,unit,_]),
   append(C, E, A).
```

Program 3bis. mandarin.ii.swipl

Notice that this program incorporates, as it must, `doublecolon.swipl`. To try out this new program, consult it and enter the following:

```
mandarin([jiu,shi,jiu],_, ME,_).
```

It will respond with

```
ME = (_G444::_G444 is 9*10+9)
```

(except that it might display a different number after `_G`). Underscore `G` followed by a numeral is PROLOG's abstruse way of displaying a variable. In this case, our referential variable `R`. So what the preceding response is telling us is that the meaning of `[jiu,shi,jiu]` is the procedure triggered by the following referential query:

```
(_G444::_G444 is 9*10+9)
```

Now, if we were to enter this query (followed by a period), we would get

```
_G444 = 99.
```

This tells us that the *reference* of `[jiu, shi, jiu]`, as opposed to its *meaning*, is the number 99.

So `mandarin.ii.swipl` tells us that the meaning paired with the Mandarin sound `[jiu, shi, jiu]` is a procedure for the computation of number 99. It is the procedure that multiplies 9 by 10 and adds 9 to the result. It should be pointed out that thus program has exactly the same complexity as the original mandarin program.

[In Section 11 of Chapter 4]

- In the NOUNS column: Replace *rise* by *teeth*.
- In the VERBS column: Replace *rise* by *teethe*.
- In the first box thereafter, line 4: Replace `[r,a,j,s]` by `[t,i,th]`

[This is a new section for Chapter 5. It should go right before the Section called "Conclusion". All subsequent sections in that chapter should be renumbered accordingly. And corresponding changes are necessary for the Table of Contents, including its page numbers].

6. Revisiting Homorganicity

We must admit that the definition of homorganicity presented in Section 2 above is unprincipled. For there we said that two phones were homorganic if they were specified the same way for labiality, dentality, alveolarity, palatality and velarity. But nowhere was it said that these features are just the ones that specify point of articulation. So the fact that they do, must be regarded as an accident.

Here's a better account. First we state that the properties *labial*, *dental*, *alveolar*, *palatal*, and *velar* indeed describe place of articulation. In other words, we state that *place description* —or, more simply, *place*— is a ***metaproperty*** of *labial*, *dental*, *alveolar*, *palatal*, and *velar* (see Program 6bis, `metaproperties.swipl`):

```
property(lab).
property(dnt).
property(alv).
property(pal).
property(vel).

place(lab).
place(dnt).
place(alv).
place(pal).
place(vel).
```

Next we say that *A* and *B* are ***congruent*** relative to *C* just in case *C* is a property that applies to *A* if and only if it applies to *B* as well. Now, one might think that this can be done simply by saying

```
congruent(A,B,C):-
          property(C),
          C(A) <=> C(B).
```

Unfortunately, this is not something we can say in PROLOG, as the second condition of this clause has, inadmissibly, two occurrences of a variable —in this case `C`— as predicates. The good news is that this can be repaired. What we can do is rewrite `C(A)` and `C(B)` as lists, and then invoke these lists by appealing to the opaquely-named universal (or `univ`) relation. The `univ` relation is a one-to-one relation that holds between a statement of the form *P(A1, A2, ..., AN)* and a list of the form *[P, A1, A2, ... AN]* consisting of the predicate of the statement followed by its arguments. Thus, since the `univ` relation is written as the infix operator `=..` we have that

`X =.. Y` holds if and only if `X` is a statement and `Y` is a list having the predicate of `X` as its head and the arguments of `X`, in order, as the elements of its tail.

The universal relation allows us to cast the illegitimate condition above into three legitimate conditions as follows. First we say what it means for two things *A* and *B* to be congruent relative to *C*:

```
congruent(A,B,C):-
          property(C),
          E =.. [C,A],
          F =.. [C,B],
          E <=> F.
```

So two things are congruent relative to a property if both things have the property or if both things lack the property. We thus exclude the possibility that one thing has the property and the other one does not.

Next we deem two things *congruent relative to a list* if and only if the two things in question are congruent relative to each and every member of the list. This we need to say in two clauses: one for empty lists of properties (see the first line of code below), and one for nonempty lists of properties (see the rest of the code):

```
congruent(_, _, []).

congruent(A, B, [C|D]):-
```

```
        congruent(A, B, C),
        congruent(A, B, D).
```

It should be clear that the last three lines of code check whether `A` and `B` are congruent relative to a list `[C|D]`of items by checking whether `A` and `B` are congruent with each item of said list, taken from left to right, one at a time.

Be that as it may, the last three clauses we have discussed constitute `congruence.swipl` (see Program 41bis of Appendix H).

We are finally in a position to improve on our definition of homorganicity. Homorganicity can now be defined, simply, as the congruence of two phones relative to the list of properties that describe place of articulation. In other words as *the congruence of two phones relative to place of articulation*:

```
homorganic(A,B):-
      phone(A),
      phone(B),
      findall(X, place(X), C),
      congruent(A, B, C).
```

Interestingly, although principledness has improved dramatically in this definition, efficiency remains essentially the same as before (it turns out to be quadratic using append but linear using the stroke notation instead). See Program 33bis of Appendix H (`newhomorganic.swipl`), which uses the stroke notation instead of append.

We should add that *place* is not the only metaproperty we might need. We might also need metaproperties for *manner* (for assimilation in manner of articulation), *aperture* (stem vowels of second-person Portugese verbs may take on the values for both HI and LO of deleted theme vowels), and *laryngeal* (word-final consonants in Korean need to be negatively marked for two features that describe the state of the larynx: TENSE and ASPIRATED). See van Oostendorp (2005).

[The paragraph below should go in Chapter 6, Section 6. There it should go between the paragraph that discusses example (23) and the paragraph that begins "We have included all our statements"]

In fact, even English allows proper names to be specified. Back in Section 3.3 we encountered the sentence *A Robert Jones is here to see you*. There we pointed out that this sentence means that a bearer of the name *Robert Jones* is here to see you (as predicted by the metalinguistic version of indirect referentialism). Here we will point out that it contains a proper name *Robert Jones* preceded by the specifier *a* (as predicted by its assignment to the category of phrases).

[The material below should be added to Appendix H. It should be ordered as appropriate, i.e. PROGRAM 3bis should come right after PROGRAM 3; PROGRAM 6bis should come right after PROGRAM 6, and so on.]

• PROGRAM 3bis: `mandarin.ii.swipl`

```
:- ['doublecolon.swipl'].

mandarin([yi ], numeral, R :: R is 1, [number, unit, first]).
mandarin([er ], numeral, R :: R is 2, [number, unit, nonfirst]).
mandarin([san], numeral, R :: R is 3, [number, unit, nonfirst]).
mandarin([si ], numeral, R :: R is 4, [number, unit, nonfirst]).
mandarin([wu ], numeral, R :: R is 5, [number, unit, nonfirst]).
mandarin([liu], numeral, R :: R is 6, [number, unit, nonfirst]).
mandarin([qi ], numeral, R :: R is 7, [number, unit, nonfirst]).
mandarin([ba ], numeral, R :: R is 8, [number, unit, nonfirst]).
mandarin([jiu], numeral, R :: R is 9, [number, unit, nonfirst]).
mandarin([shi], numeral, R :: R is 10,[number, ten,  first]).

mandarin(A, numeral, R :: R is N1 * N2,[number, ten, nonfirst]):-
       mandarin(C, numeral, R1 :: R1 is N1,[number, unit,nonfirst]),
       mandarin(E, numeral, R2 :: R2 is N2,[number, ten, first]),
       append(C, E, A).

mandarin(A, numeral, R :: R is N1 + N2,[number]):-
       mandarin(C, numeral, R1 :: R1 is N1,[_,ten,_]),
       mandarin(E, numeral, R2 :: R2 is N2,[_,unit,_]),
       append(C, E, A).
```

• PROGRAM 6bis: `metaproperties.swipl`

```
:- ['fullproperties.swipl'].


property(lab).
property(dnt).
property(alv).
property(pal).
property(vel).
property(cns).
property(cnt).

place(lab).
place(dnt).
place(alv).
place(pal).
place(vel).

manner(cns).
manner(cnt).
```

• **PROGRAM 33bis: `newhomorganic.swipl`**

```
:- ['congruence.swipl'].

underenglish([P1,P2,P3,P4],[adjective]):-
          phone(P1), not(voi(P1)), not(dnt(P1)), lab(P1),
          phone(P2), mid(P2), not(bck(P2)), not(ctr(P2)),
          phone(P3), snt(P3), not(nas(P3)), not(alv(P3)), pal(P3),
          phone(P4), not(snt(P4)), voi(P4), not(cnt(P4)), alv(P4).

underenglish([P1,P2,P3],[adjective]):-
          phone(P1), not(voi(P1)), dnt(P1), lab(P1),
          phone(P2), mid(P2), not(bck(P2)), not(ctr(P2)),
          phone(P3), not(nas(P3)), alv(P3), pal(P3).

underenglish([P1,P2,P3,P4,P5,P6,P7],[adjective]):-
          phone(P1), not(voi(P1)), cnt(P1), cor(P1), not(sib(P1)),
          phone(P2), hih(P2), not(bck(P2)), not(tns(P2)),
          phone(P3), nas(P3),
          phone(P4), not(voi(P4)), vel(P4),
          phone(P5), not(cns(P5)), not(str(P5)),
          phone(P6), not(snt(P6)), voi(P6), not(cnt(P6)), lab(P6),
          phone(P7), snt(P7), not(nas(P7)), alv(P7), not(pal(P7)).

underenglish([P1,P2,P3,P4,P5,P6],[adjective]):-
          phone(P1), not(voi(P1)), not(cnt(P1)), alv(P1),
          phone(P2), hih(P2), not(bck(P2)), not(tns(P2)),
          phone(P3), not(voi(P3)), not(dnt(P3)), lab(P3),
          phone(P4), hih(P4), not(bck(P4)), not(tns(P4)),
          phone(P5), not(voi(P5)), vel(P5),
          phone(P6), snt(P6), not(nas(P6)), alv(P6), not(pal(P6)).

underenglish([P1,P2,P3,P4,P5,P6,P7,P8], [adjective]):-
          phone(P1), not(nas(P1)), alv(P1), pal(P1),
          phone(P2), not(bck(P2)), tns(P2),
          phone(P3), snt(P3), not(nas(P3)), alv(P3), not(pal(P3)),
          phone(P4), mid(P4), not(bck(P4)), not(ctr(P4)),
          phone(P5), snt(P5), not(nas(P5)), not(alv(P5)), pal(P5),
          phone(P6), not(voi(P6)), not(cnt(P6)), alv(P6),
          phone(P7), hih(P7), not(bck(P7)), not(tns(P7)),
          phone(P8), not(snt(P8)), voi(P8), not(cnt(P8)), alv(P8).

underenglish([P1,P2,P3,P4,P5,P6],[adjective]):-
          phone(P1), not(voi(P1)), not(cnt(P1)), pal(P1),
          phone(P2), mid(P2), not(bck(P2)), not(ctr(P2)),
          phone(P3), snt(P3), not(nas(P3)), not(alv(P3)), pal(P3),
          phone(P4), nas(P4),
          phone(P5), not(snt(P5)), voi(P5), cnt(P5), pal(P5),
          phone(P6), not(snt(P6)), voi(P6), not(cnt(P6)), alv(P6).

underenglish([P1,P2,P3,P4],[adjective]):-
          phone(P1), not(voi(P1)), vel(P1),
          phone(P2), snt(P2), not(nas(P2)), alv(P2), not(pal(P2)),
          phone(P3), not(bck(P3)), tns(P3),
          phone(P4), not(nas(P4)), alv(P4), pal(P4).
```

```
underenglish([P1,P2],[affix,negative]):-
          phone(P1),ctr(P1),str(P1),
          phone(P2),nas(P2).

underenglish([A,B|C],[adjective,negative]):-
          underenglish([A,B],[affix,negative]),
          underenglish(C,[adjective]).

english(A,B):-
          underenglish(A,B),
          (nas(N), cns(C), nextto(N,C,A)) => homorganic(N,C).

homorganic(A,B):-
          phone(A),
          phone(B),
          findall(X, place(X), C),
          congruent(A, B, C).
```

• **PROGRAM 41bis: `congruence.swipl`**

```
:- ['metaproperties.swipl'].
:- ['entailment.swipl'].


congruent(A, B, [C|D]):-
          congruent(A, B, C),
          congruent(A, B, D).

congruent(A,B,C):-
          property(C),
          E =.. [C,A],
          F =.. [C,B],
          E <=> F.

congruent(_, _, []).
```

• PROGRAM 26: replace english([r,a,j,s], noun) by english([t,i,th], noun)

[The material below should be added to the References section in its correct alphabetical position]

van Oostendorp, Marc (unpublished manuscript) "Feature Geometry". October 4, 2005.

Appendix F

THE FUNDAMENTAL PROPERTIES OF ENGLISH PHONES

	p	b	m	t	d	n	k	g	ŋ	f	v	θ	ð	s	z	ʃ	ʒ	č	ǰ	l	ɹ	j	w	h	i	ɪ	e	æ	u	ʊ	o	ɑ	ə	ʌ
cns	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	–	–	–	–	–	–	–	–	–	–
snt	–	–	+	–	–	+	–	–	+	–	–	–	–	–	–	–	–	–	–	+	+	+	+	–	+	+	+	+	+	+	+	+	+	+
nas	–	–	+	–	–	+	–	–	+	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
voi	–	+	+	–	+	+	–	+	+	–	+	–	+	–	+	–	+	–	+	+	+	+	+	–	+	+	+	+	+	+	+	+	+	+
cnt	–	–	–	–	–	–	–	–	–	+	+	+	+	+	+	+	+	–	–	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
lab	+	+	+	–	–	–	–	–	–	+	+	–	–	–	–	–	–	–	–	–	–	–	+	–	–	–	–	–	–	–	–	–	–	–
alv	–	–	–	+	+	+	–	–	–	–	–	–	–	+	+	–	–	–	–	+	+	–	–	–	–	–	–	–	–	–	–	–	–	–
pal	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	+	+	+	+	–	+	+	–	–	–	–	–	–	–	–	–	–	–	–
ant	+	+	+	+	+	+	–	–	–	+	+	+	+	+	+	–	–	–	–	+	+	–	–	–	–	–	–	–	–	–	–	–	–	–
vel	–	–	–	–	–	–	+	+	+	–	–	–	–	–	–	–	–	–	–	–	–	–	+	–	–	–	–	–	–	–	–	–	–	–
cor	–	–	–	+	+	+	–	–	–	–	–	+	+	+	+	+	+	+	+	+	+	+	–	–	–	–	–	–	–	–	–	–	–	–
sib	–	–	–	–	–	–	–	–	–	–	–	–	–	+	+	+	+	+	+	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
hih	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	+	+	–	–	+	+	–	–	–	–
mid	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	+	–	–	–	+	–	+	+
low	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	+	–	–	–	+	–	–
bck	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	+	+	+	+	–	–
ctr	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	+	+
tns	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	+	–	–	–	+	–	–	–	–	–
str	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	+	+	+	+	+	+	+	+	–	+