University of Auckland

07.363

Functional and Logic Programming

Christian Collberg

October 7, 1994

Assigments

Copyright © 1994 Christian Collberg

Speech Synthesis I

Main Program:

```
init :- res_open('English Sounds').

talk :-
   ask(['Sentence'], W), mapwords(W, S),
   match([],S,R,P), play(P).
```

Map chars to atoms:

```
mapwords([W|WL],S) :-
   name(W, N), mapchars(N, S1),
   mapwords(WL, S2), append(S1,[blank|S2], S).
mapwords([],[]).

mapchars([C|N], [C1|S]) :-
   mapchar([C], C1), mapchars(N, S).
mapchars([],[]).

mapchar(X, Y) :-
   X>="A", X=<"Z", X1 is X+32, name(Y, [X1]).
mapchar(X, Y) :-
   ((X>="a", X=<"z") ; (X>="0", X=<"9")),
   name(Y, X).
```

Slide 16–1

Speech Synthesis II

```
mapchar("!", excl_mark).  mapchar("?", q_mark).
mapchar(".", full_stop).  mapchar(",", comma).
mapchar("-", hyphen).  mapchar(";", s_colon).
mapchar("$", dollar).

mapchar(X, X).
```

Play sounds:

```
play([silent|T]) :-play(T).
play([X|T]) :-
   sound(X,S), snd_play(S), play(T).
play([]).

% Vowel Phonemes
sound(iy, 29808).  % bEEt
sound(ey,9946).  % gAte
% Consonant Phonemes
sound(p, 21063).  % Pack
sound(t, 13922).  % Time

sound(pause,25590).
sound(silent,0).
```

Slide 16–2

Speech Synthesis III

Main matching routine:

```
match(Left, Right, [rule(C,N)|U], Out) :-
   [C|_] = Right,
   rule(C, N, LCxt, Match, RCxt, Phs),
   append(Match, R, Right),
   prefix_match(RCxt, R),
   suffix_match(LCxt, Left),
   !,
   append(Left, Match, L),
   match(L, R, U, O),
   append(Phs, O, Out).
match(_,[],[],[]).
```

Letter qualities:

```
vowel(a).  vowel(o).  ··· vowel(i).
vowelF(e).  vowelF(i).  vowelF(y).
cons(b).  cons(c).  cons(d).  ··· cons(z).
consV(b).  consV(d).  ··· consV(z).
```

Slide 16–3

Speech Synthesis IV

Prefix matching:

```
% vowel  -- 1 or more vowels
% cons0  -- 0 or more consonants
% cons1  -- 1 consonant
% consV  -- 1 voiced consonant
% suffix -- ed, ing, ...
% vowelF -- Front vowel
% any    -- anything
% none   -- nothing

prefix_match(any,_).
prefix_match(none, []).
prefix_match(none,blank).
prefix_match(none, [blank|_]).
% One voiced consonant
prefix_match([consv|S],[C|R]) :-
   consV(C), prefix_match(S, R).
% One consonant
prefix_match([cons1|S],[C|R]) :-
   cons(C), prefix_match(S, R).
```

Slide 16–4

Speech Synthesis V

```
% One or more vowels
prefix_match([vowel|S],[V|R]) :-
   vowel(V), (prefix_match(S, R);
         prefix_match([vowel|S],R)).
% Zero or more consonants
prefix_match([cons0|S],[C|R]) :-
   (cons(C),
      (prefix_match(S,R);
         prefix_match([cons0|S], R)));
   prefix_match(S, [C|R]). % Not a consonant
% One front vowel.
prefix_match([front|S],[V|R]) :-
   vowelF(V), prefix_match(S, R).

% Common suffixes
prefix_match([suffix|S],[i,n,g|R]) :-
   prefix_match(S,R).
prefix_match([suffix|S],[e,l,y|R]) :-
   prefix_match(S,R).
```

Slide 16–5

Speech Synthesis VI

```
prefix_match([suffix|S],[e,r|R]) :-
   prefix_match(S,R).
prefix_match([suffix|S],[e,s|R]) :-
   prefix_match(S,R).
prefix_match([suffix|S],[e,d|R]) :-
   prefix_match(S,R).
prefix_match([suffix|S],[e|R]) :-
   prefix_match(S,R).

% One letter
prefix_match([X|S],[X|R]) :-
   prefix_match(S, R).
% End of the context
prefix_match([],_).
```

Suffix matching:

```
suffix_match(any,_).
suffix_match(none, []).
suffix_match(none, Word) :-
   reverse(Word, W), prefix_match(none,W).
```

Slide 16–6

Speech Synthesis VII

```
suffix_match([C], Word) :-
   reverse(Word, W),
   prefix_match([C],W).
suffix_match([C1,C2], Word) :-
   reverse(Word, W),
   prefix_match([C2,C1],W).
suffix_match(Context, Word) :-
   reverse(Context, C),
   reverse(Word, W),
   prefix_match(C,W).

% Rule-letter, number, Left Ctxt,
% MatchChars, Right Ctxt, Phonemes
rule(blank, 1, any, [blank], any, [pause]).
rule(a, 1, any, [a], none, [ax]).
rule(a, 2, none, [a,r,e], none, [aa,r]).
rule(a, 3, none, [a,r], [o], [ax,r]).
rule(a, 4, any, [a,r], [vowel], [eh,r]).
rule(a, 5, [cons1], [a,s], [vowel], [ey,s]).
```

Slide 16–7

Ass 3 – Stable Marriages I

Input database:

```
prefer(avraham, man,
   [chana, tamar, zvia, ruth, sarah]).
prefer(binyamin, man,
   [zvia, chana, ruth, sarah, tamar]).
prefer(chaim, man,
   [chana, ruth, tamar, sarah, zvia]).
prefer(david, man,
   [zvia, ruth, chana, sarah, tamar]).
prefer(elazar, man,
   [tamar, ruth, chana, zvia, sarah]).
prefer(zvia, woman,
   [elazar, avraham, david, binyamin, chaim]).
prefer(chana, woman,
   [david, elazar, binyamin, avraham, chaim]).
prefer(ruth, woman,
   [avraham, david, binyamin, chaim, elazar]).
prefer(sarah, woman,
   [chaim, binyamin, david, avraham, elazar]).
prefer(tamar, woman,
   [david, binyamin, chaim, elazar, avraham]).
```

Slide 16–8

Ass 3 – Stable Marriages II

Main routine:

```
go :-
   build_people(ML, WL),
   start_picture(ML, WL),
   gen(ML, WL, [], L),
   not(unstable(L)),
   link_list(ML, WL, L, or),
   show(L),
   link_list(ML, WL, L, erase),
   fail.
go.

build_people(ML, WL) :-
   setof(M, X^prefer(M, man, X), ML),
   setof(W, X^prefer(W, woman, X), WL).
```

Slide 16–9

Ass 3 – Stable Marriages III

Generate:

```
gen([A|M1], W, In, Out) :-
   delete(B, W, W1),
   gen(M1, W1, [m(A,B)|In], Out).
gen([],[],L,L).

delete(A, [A|L], L).
delete(A, [X|L], [X|L1]) :- delete(A, L, L1).
```

Test:

```
% A prefers B to C.
pref(A, B, C) :-
   prefer(A, _, L),
   append(_, [B|S], L), !, member(C, S), !.

unstable(L) :-
   append(_, [A|R], L), member(B, R),
   (is_unstable(A,B); is_unstable(B,A)).

is_unstable(m(A,Y), m(B,X)) :-
   pref(A, X, Y), pref(X, A, B).
```

Slide 16–10

Ass 3 – Stable Marriages IV

Textual Display:

```
show(L) :-
   atomize(L, L1),
   scroll_menu(['Stable:'], L1,[],G), !.
show(_) :- abort.

atomize([],[]).
atomize([m(A,B)|L],[F|L1]) :-
   name(A, A1), name(B, B1),
   append(A1,"-", F1),
   append(F1,B1,F2), name(F, F2),
   atomize(L, L1).
```

Graphic Display:

```
start_picture(M, W) :-
   grafix, del_all(usr),
   display_list(M, 60), display_list(W, 300).

pos(M, L, P) :-
   append(V,[M|_],L), length(V, R), P is R+1.
```

Slide 16–11

Ass 3 – Stable Marriages V

```
link_it(ML, WL, M, W, D) :-
   pos(M,ML,MP), pos(W,WL,WP),
   Mpos is 30 + (MP-1) * 40,
   Wpos is 30 + (WP-1) * 40,
   add_pic(penmode(D, pensize(2,2,
      pointer((Mpos,165), (Wpos,295), both)))).

link_list(_, _, [], _).
link_list(ML, WL, [m(M,W)|R], D) :-
   link_it(ML, WL, M, W, D),
   link_list(ML, WL, R, D).

display_list(L, C) :- display_column(L, 20, C).

display_column([T|L], X, Y) :-
   add_pic(textbox('times',
         14,1,X,Y,20,100,1, T,on)),
   X1 is X + 40, display_column(L, X1, Y).
display_column([],_, _).
```

Slide 16–12

Setof/Bagof Revisited

```
age(peter, 7).  age(ann, 5).
age(pat, 8).  age(tom.  5).

?- bagof(Child, age(Child, Age), L).
   Age=7, L = [peter] ;
   Age=5, L = [ann, tom] ;
   Age=8, L = [pat] ;
   no
```

- `A^` — Any `A`, as long as one exists.

```
?- bagof(C, Age^age(C, Age), L).
   L = [peter, ann, pat, tom]

?- setof(A, Child^age(Child, A), L).
   L = [5, 7, 8]
```

Slide 16–13

Ass 3 – Rational Arithmetic I

Operator declarations:

```
:- op(1000, xfx, isr).
:- op(100, fy, ~).
:- op(10, xfx, \).
```

Main routine:

```
comp(Z, X, Y, A, B, C, D, Num, Den) :-
   X1 isr X,
   Y1 isr Y,
   rat(X1, A\B),
   rat(Y1, C\D),
   Z1 is Num,
   Z2 is Den,
   simp(Z1\Z2, Z3\Z4),
   rat(Z, Z3\Z4).
```

Auxiliary predicate:

```
rat(A, A\1) :- number(A).
rat(A \ B, A \ B).
```

Slide 16–14

Ass 3 – Rational Arithmetic II

Arithmetic:

```
X isr X :- number(X).

Z isr X \ Y :-
   comp(Z, X, Y, A, B, C, D, A*D, B*C).

Z isr X * Y :-
   comp(Z, X, Y, A, B, C, D, A*C, B*D).

Z isr X + Y :-
   comp(Z, X, Y, A, B, C, D, A*D+C*B, B*D).

Z isr X - Y :-
   comp(Z, X, Y, A, B, C, D, A*D-C*B, B*D).

Z isr X / Y :-
   comp(Z, X, Y, A, B, C, D, A*D, B*C),

Z isr ~ X :-
   X1 isr X, rat(X, A \ B),
   simp(A \ B, A1  B1), rat(Z, B1 \ A1).
```

Slide 16–15

Ass 3 – Rational Arithmetic III

Simplification

```
simp(A\B, C\D) :-
   number(A),
   number(B),
   gcd(A, B, S),
   C is A / S,
   D is B / S.
simp(A, A).

gcd(X, X, X).
gcd(X, Y, D) :-
   X<Y,
   Y1 is Y-X,
   gcd(X, Y1, D).
gcd(X, Y, D) :-
   Y < X,
   gcd(Y, X, D).
```