**Relations list**

all relations have 1 or 2 kinds of arguments (not more).

S,D symmetric, p gets transferred. Similar, Identical

AND,OR,XOR,NOT symmetric, some p rules apply to compound And, Or, excl or,Not

C asymetric, p trasfer from class children Class

F asymetric, implied C relation: C(a,F(a,b)) Feature

Q asymetric, implied C relation: C(b,Q(a,b)), g transfer Quantifier

A asymetric, implied C relation: C(b,A(a,b)) Action

I asymetric, implied C relation: C(a,I(a,b)) Impact (on object)

P asymetric, no C relation, no p transfer Part or Possession

R asymetric, no C relation, no p transfer Relative (space etc)

T asymetric, no C relation, no p transfer Time

M ordered, no C relation, no p transfer More

IM asymetric, no C relation, yes p transfer Implication

N asymetric, no C relation, yes p transfer Necessary condition

V asymetric, no C relation, no p transfer Relevant

**Mapping example using D-relation**

*Joe and Mary are playing in the sandbox. He is her brother.*

KB tartalom:

IM(F(person,male), NOT(F(person,female)))

IM(F(person,female), NOT(F(person,male)))

Vagy valami mas kifejezese annak hogy ezek egymast kizarjak,

peldaul F(person, AND(male, female)) p=0 ezt tomoren kifejezheti

vagy AND(F(person,male),F(person,female))p=0.

C(Joe, person)

F(Joe, male)

C(Mary, person)

F(Mary, female)

C(he, person)

F(he, male)

C(brother, person)

F(brother, male)

D(he, %1) vagy az IM-reasoning segitsegevel: IM(he, D(he,%1))

N(D(he,%1), A(%1,%2)) - N a “condition”, feltetel relacio, D(he,%1) biztosan nem igaz ha A(%1,%2) nem igaz.

Ez a felhasznalando mapping rule.

Input alakulasa a WM-ben:

Joe F(Joe, female)p=0 and Mary F(Mary, male)p=0 AND(Joe,Mary) … A(Joe,play) A(Mary, play)… he **H11** D(he, Mary) C(he, person) F(he, female) F(Mary, male) **H11end** **H12** D(he, Joe) C(he, person) F(he, male) F(Joe, male) **H12end H13 H13end H12** is her brother Q(her, brother) D(he, Q(her, brother)) …

1. Joe utan C-reasoning es IM-reasoning eredmenye hogy F(Joe,female)p=0. Hasonloan F(Mary,male)p=0.
2. H11 helyen Map concept fut, “he” gyerekei kozott megtalalja hogy D(he,%1). Megtalalja D(he,%1) gyerekekent hogy N(D(he,%1), A(%1,%2)). Emiatt keresni kezd visszafele egy A() conceptet, megtalalja hogy A(Mary,play). Felveszi a H11 elagazast, es WM-be a mappinget D(he, Mary).
3. Most kovetkezik a reasoning (C, IM, D reasoning) amibol most D-reasoning mukodik, ebbol felveszi a WM-be hogy F(he, female)p=1 es F(Mary, male)p=1.
4. Most fut update lm value of assumption H11-re, osszehasonlitja F(Mary,male)p=1 a korabbi F(Mary,male)p=0 concepttel, ami negative lm update-et eredmenyez, mondjuk lm=-100, ezt a H11-hez rendelve tarolja.
5. H12-ben lm update pozitiv, +100, ezt H12-hoz rendelve tarolja.
6. H13 lm-je nulla.
7. Most fut keep top mapping assumption ami ezek kozul valaszt: H11 r=100 lm=-100 H12 r-100 lm=100 H13 r=0 lm=0 , kivalasztja H12-t, a masik kettot killed flaggel latja el.
8. Veszi a kovetkezo szot, is, mar csak a H12 agban.

**M0 main file**

Read an ini file and set global parameters.

Initialize the word list, the knowledgebase, and the working store.

Populate mapping between concept relation type letters and numeric codes.

Call M1.

Call M5: Save working store finally, to save the results in a file.

\*\*\* defaults \*\*\*

Concept size: 1000

Word size: 1000

Default r: 50

Default p: 100

Default c: 100

\*\*\*

\*\*\* concept code mapping \*\*\*

W: 0 //word

S: 1 //similar

D: 2 //identical

C: 3 //class 1:member 2:class

F: 4 //feature 1:entity 2:feature

Q: 5 //quantifier 1:quantifier 2:entity

A: 6 //action 1:actor 2:action

I: 7 //impact 1:actor 2:impacted object

R: 8 //relative 1:speficier 2:entity e.g. with, hand

T: 9 //time 1:entity(action) 2:time-expression

P: 10 //part, possessing 1:owned entity/part 2:owner/whole

M: 11 //more, ordered list. 1:feature1 2:feature2 …

IM: 12 //implication 1:reason 2:consequence

N: 13 //necessary condition 1:condition 2:consequence

V: 14 //relevance

AND: 15 //and

OR: 16 //or

NOT 17 //not

\*\*\*

**M1 Mentalese input**

Input: file name with mentalese sentences.

Read input (from file), later receive the output of mentalese translator.

See example input.

For each mentalese sentence

* Call M5: store concepts in working store.

**M2 Word list**

Add new word. Input: word.

* *Determine if it is a word. (comes later)*
* If word, add to word list and knowledgebase:
  + append word to word list. (fixed size)
  + Call M4 Create empty concept, with parent pointer to the word (with type=0)
  + Set pointer in the word to the new concept.

Find word. Input: word.

* Search word in word list.
* Return pointer or not found flag.

Get number of meanings. Input: pointer to word.

* Return number of meanings.

**M3 Concept**

Create empty concept. Input: store (knowledgebase/working), number of parents, pointers to parents, and concept type. (number of parents=1 for a new word.)

* initialize new concept.
* Set number of parents.
* Set parent pointers.
* Set concept relation type.
* Return pointer to new concept.

Set r (input: pointer to concept, r)

Set p

Set c

Store word in WM

Store compound concept in WM Input: concept

* + - M3: Create next empty concept in working store, with pointers to parents (use a mapping between concept type letters in input, and concept codes)
    - M3: set r, p, c to default values
    - For all parent concepts, set pointer to this new concept as child
    - If parents have link to KB, call M4: search concept in KB based on parents
      * If success, set KB link - that will show to the same concept in the KB.

Start assumption (or hypothesis) – this is to fork the WM when more than one possibility needs to be evaluated against each other. Like for word sense disambiguation and for mapping.

Input: new assumption or continuation; last concept in WM

Suspend assumption – to stop evaluation in this assumption branch.

\*\*\* concept specification \*\*\*

* Fields
  + Number of fields
  + Concept relation type
  + r, relevance
  + p, probability
  + c, confidence
* link to knowledgebase
  + pointer to the same concept in the knowledgebase
* parents
  + number of parents
  + pointers to parents
* children
  + number of children
  + pointers to children
* extension
  + pointer to the extension of this concept

\*\*\*

**M4 Knowledgebase**

Store concepts in long term memory.

Create empty concept in knowledgebase. Input: number of parents, pointers to parents, and concept type. (number of parents=1 for a new word, type=0 for word.)

* Call M3: create empty concept, with store=knowledgebase.

Save knowledgebase (to some automated file name)

Search concept based on parents (search needs to be done in KB+WM. If the concept is found in WM first, then continue search in KB. If both found, KB needs to be returned.)

For a compound concept in WS, it tries to find it in KB, based on parents.

Input: concept in WS.

Walk through all children of one parent in KB

Match relation type and other parent`s pointer

If matching, the concept is found, return its pointer

Stop after first match (so now a compound concept should have 1 meaning only)

Return pointer in KB (WM) – **main thing is that we need to return the concept in KB or the first occurrence in WM (or we make use of the D() mapping relations, and no matter which WM occurrence is returned)**

**M5 Working store**

**Main loop of input processing:**

Store concepts in working store. Input: mentalese sentence.

1. walk through all assumptions (not being dead)
   1. take next item „a” from input (completed concept or next word), continue the assumption
   2. if not word, Store compound concept in WM (so all WM content is linked to KB if possible.)
   3. if word, generate all word meaning assumptions, walk through these
      1. Store „ai” in WM, the i-th meaning of „a”
      2. perform D-reasoning, C-reasoning, IM-reasoning
      3. generate all mapping assumptions (if any), walk through these
         1. M6: Map concept (store D-relation)
         2. perform D-reasoning, C-reasoning, IM-reasoning
         3. M6: update lm value of assumption
         4. Suspend this mapping assumption
      4. keep top mapping assumption, kill all others
   4. keep top word meaning assumption

question answering

1. walk through all assumptions (not being dead)
   1. when searching for the answer to F(Joe,?) all D(x,Joe) and D(Joe,y) need to be considered, so the search need to be extended to F(x,?) and F(y,?).
2. keep best answer

Save working store (to some automated file name)

**M6. Mapping**

Map concept input: concept „a” in WM (a word like „it” or a concept like Q(f,g) (e.g. this fox))

1. Use the KB-link of „a” (if no link, try call map concept based on synonim, EXIT)
   1. if a=Q() then arrive at b=Q(f,%1) (e.g. Q(this,%1)
2. browse through children of „a”, search for a D(Q(this,%1),%2) with r>r0 (r relevance will show the need for mapping, r0 is a global threshold) (M is the mapping relation)
   1. if none, try call map concept based on synonim
   2. EXIT
3. Set next mapping assumption branch in WM (H1, H2, ....)
   1. H1 means no mapping, suspend H1
   2. if not H1, take next D(b,%1)
      1. take next N(D(b,1%), x), condition. The mapping-rule is the N, for example x=A(%1,%2) which means the subject, mentioned previously.
         1. search for x in WM, stop if Mapping rule stop exceeded (e.g. A( , ) ) Walk through all x found:
            1. pick the %1 concept from WM, let this be d
            2. set up next mapping assumption, Hn, generate D(a,d) , store - here we originally had D(b,%1) and concluded to D(a,d)
            3. Update r-value of this assumption using update r in assumption based on N
   3. end while

Update lm value of assumption – likelihood measure; we want to increase if knowledge in a specific mapping or word sense assumption matches any other knowledge in WM/KB. Input: assumption start. (only latest start needs to be taken.)

Something like this: for all concepts within the last assumption (this will now include reasoning results plus input content as well) we search through the earlier WM content of the same branch, and KB. If found, compare p values. The result of this comparison is to be added to the lm of the assumption.

**M7. decision rules (model parameters)**

r0 global threshold for M relation can return now 0, this is to decide whether a mapping rule should be used or not.

Mapping rule stop for now can return “stop” if the search exceeds the previous sentence backwards.

update r in assumption based on N for now returns r value of the N relation. This is the way the mapping rule influences the acceptance of the mapping hipothesis.

keep top mapping assumption for now returns the top 1 or 2 by adding r+lm. This is to decide which to kill, which to keep.

keep top word meaning assumption now returns the top 1 or 2 by p+lm. This is to decide which to kill, which to keep.

keep best answer now returns the answer in the assumption with top r+lm. This is to decide which answer to choose.

compare p values of same concept now returns MIN(p1,p2). This is to find out if an assumption matches earlier knowledge.

**M8. C-reasoning**

Set up C relation with rule-type concept if the WM input is X(a,b) and we have X(a,%1) or similar in the KB then generate C(X(a,b),X(a,%1)).

Reasoning based on C Input: concept in WM, b. There are 3 cases where class for b can be found: 1 rule-type concepts wit % placeholders 2 classic C(b,c) relations 3 classes of mapped concepts.

1. if b is of form Y(a,x) then search Y(%,x) and Y(a,%) as children of y or a, because these two concepts are classes for b. let this be now c=Y(%,x).
   1. If found, then for any X(c,d) child of c, generate X(b,d) in WM. Copy over p value from X(c,d).
2. use the KB-link of “b” which shows to KB.
   1. Browse through children to find C(b,c) ( b is in class c)
   2. If found, then for any X(c,d) child of c, generate X(b,d) in WM. Copy over p value from X(c,d).
3. If b is mapped, like M(x,b), walk through mapping partners x
   1. Browse through children to find C(x,c) ( x is in class c)
   2. If found, then for any X(c,d) child of c, generate X(b,d) in WM. Copy over p value from X(c,d).

(seems that there is no need to browse through all concepts in WM to see if any of these is C(b,c). If there was such a concept, it was among children of b, and b would be either stored in KB or an earlier input in WM. In both cases the new occurrence of b is linked to it. So we can start with using the KB-link. If b has multiple occurences, then they are mapped or not. If not, they are separate instances. If mapped, maybe C-reasoning needs to be performed again, this is in 2.)

**M9. IM-reasoning**

Store IM input in WM If the mentalese input is i=IM(a,b) then store i, a and b in WM with p=0.5.

Generate concepts based on IM Input : IM-concept in WM, i=IM(a,b) ( a and b stored).

1. take „a”
2. Map „a” in WM (see mapping below) -> this may result a M(a,e) relation.
   1. if mapped, copy p value from e to a.
3. If not mapped, if „a” is not a Q-relation (then we take it being general):
   1. search „a” in KB, if found, copy p value from KB to „a”
4. take „b”
   1. let p1 be the p-value of i and p2 the p-value of „a”
   2. set p of b = MIN(p1,p2)

**M10. D-reasoning**

Reasoning based on D input:D-relation in WM