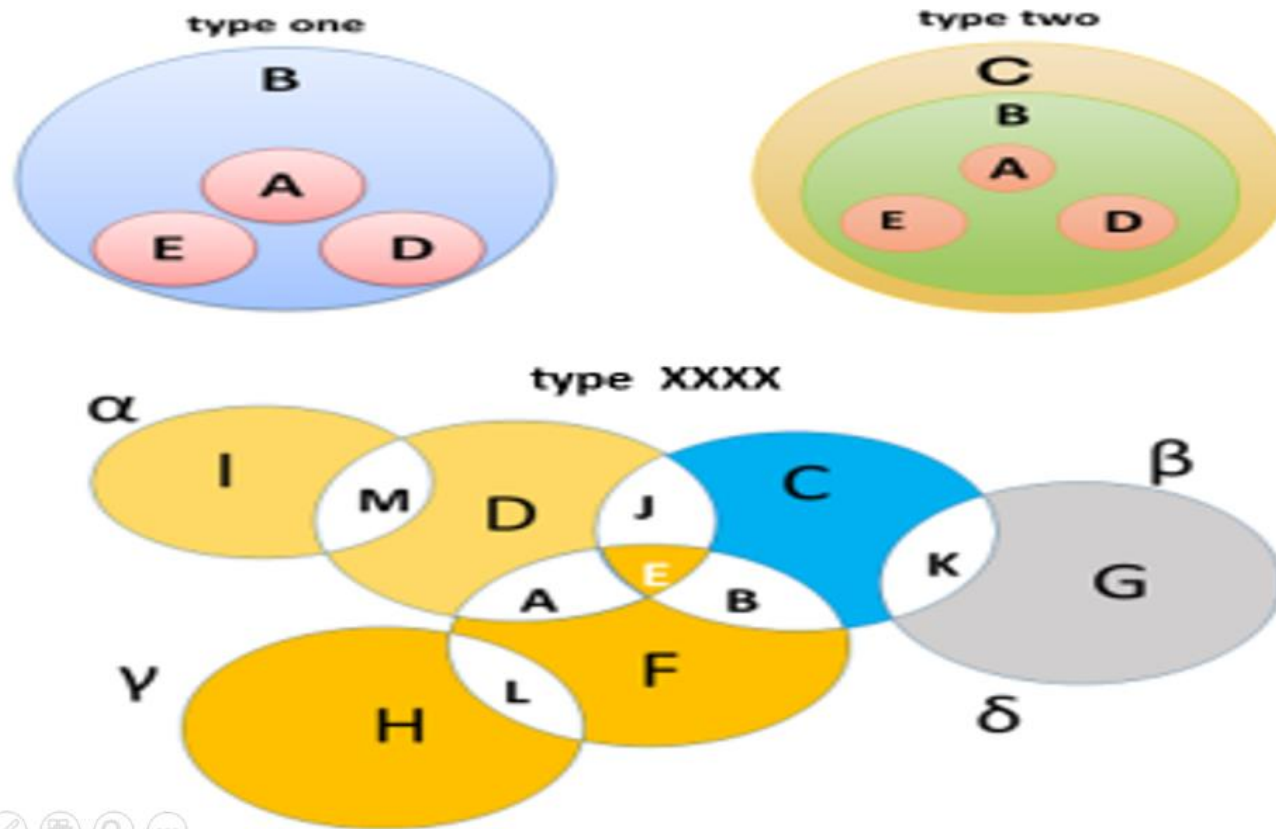


データベースを使った会話システム

www.aiplab.com 萩原良昭



Type One 会話システム

```
AIPS(hello) - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V)
7
001
how are you
I am fine, thank you.

002
who are you
I am your computer.

003
what are you
I am a computer.
|
```

TYPE ONE

What is A ?

A is B .

```
C:\Users\g\Desktop\HelloAIPS131008\HelloAIPS.exe
***** HelloAIPS *****

Hello, how are you ?

*****

AIPS-->how are you ?

I am fine, thank you.

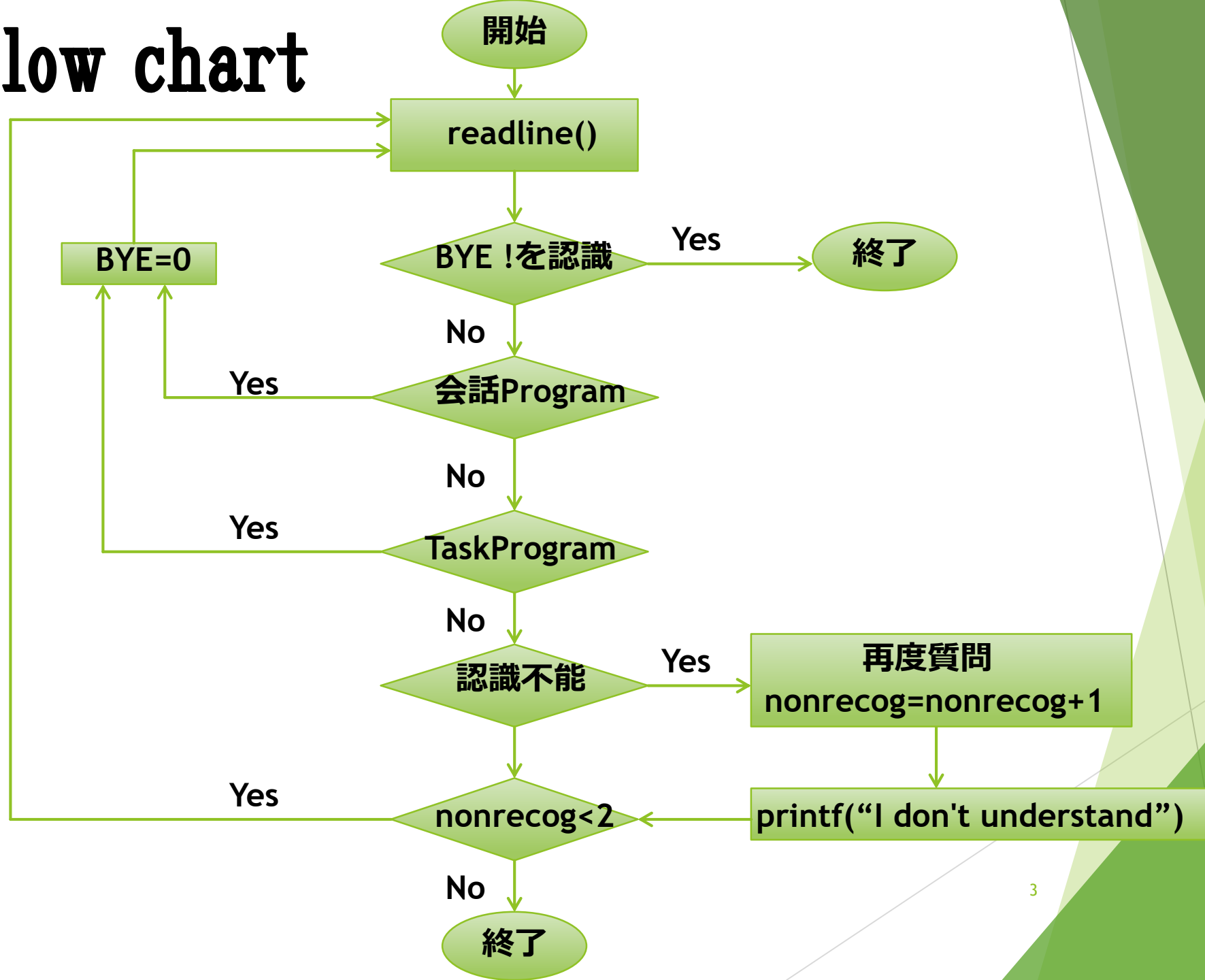
AIPS-->who are you ?

I am your computer.

AIPS-->what are you ?

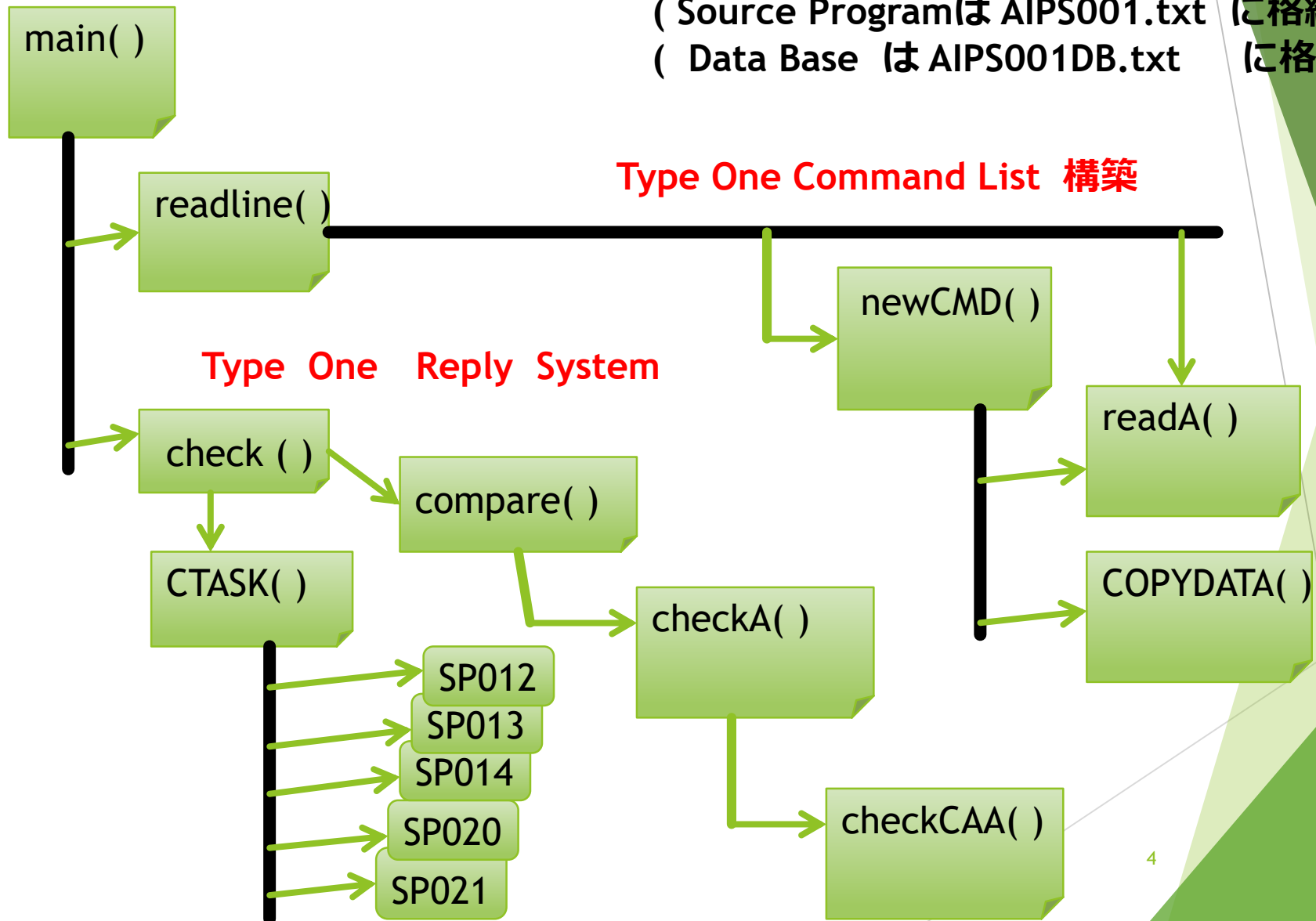
I am a computer.
```

Flow chart



Type One 会話システムProgramの構造

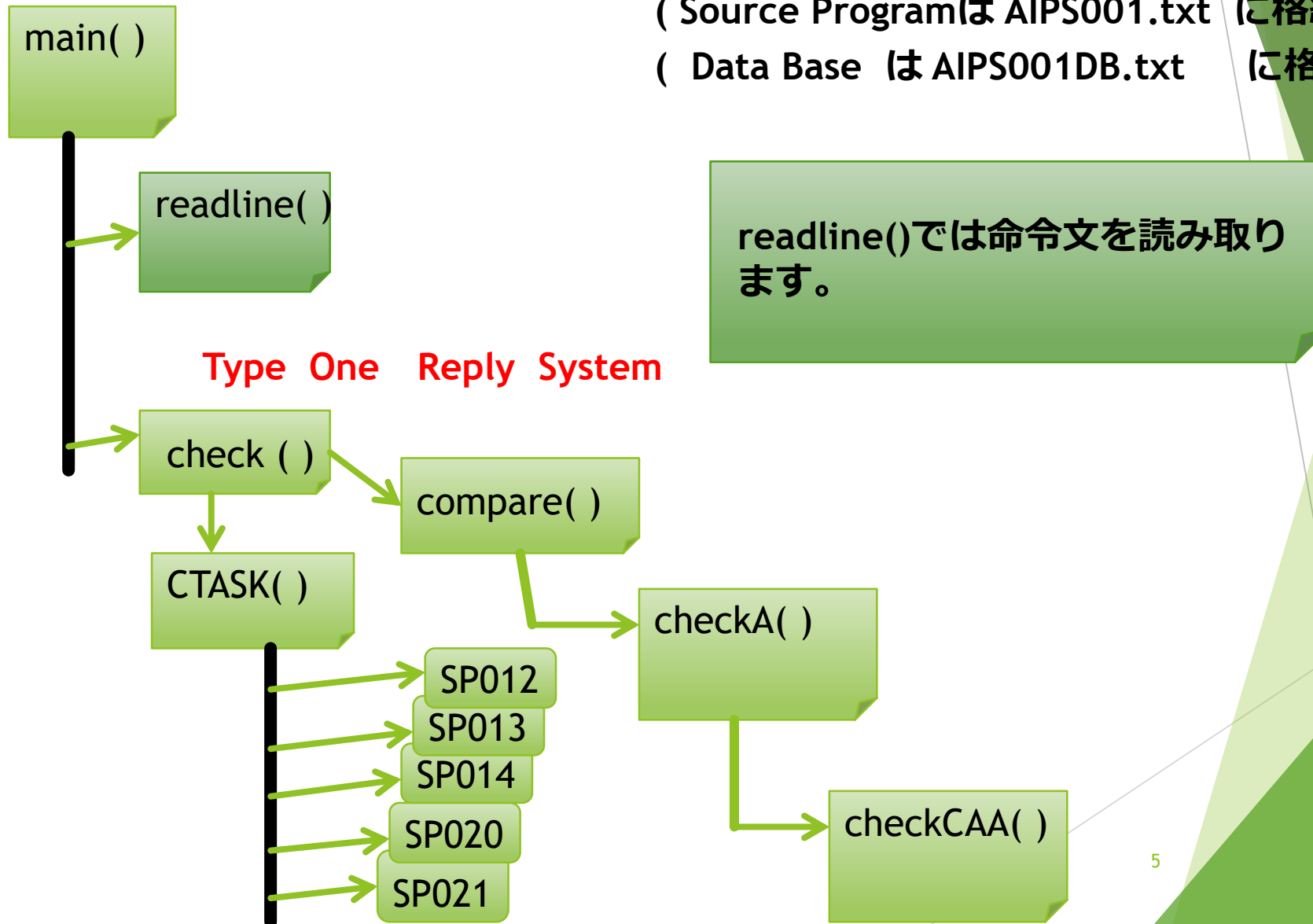
(Source Programは AIPS001.txt に格納)
(Data Base は AIPS001DB.txt に格納)



Type One 会話システムProgramの構造

(Source Programは AIPS001.txt に格納)

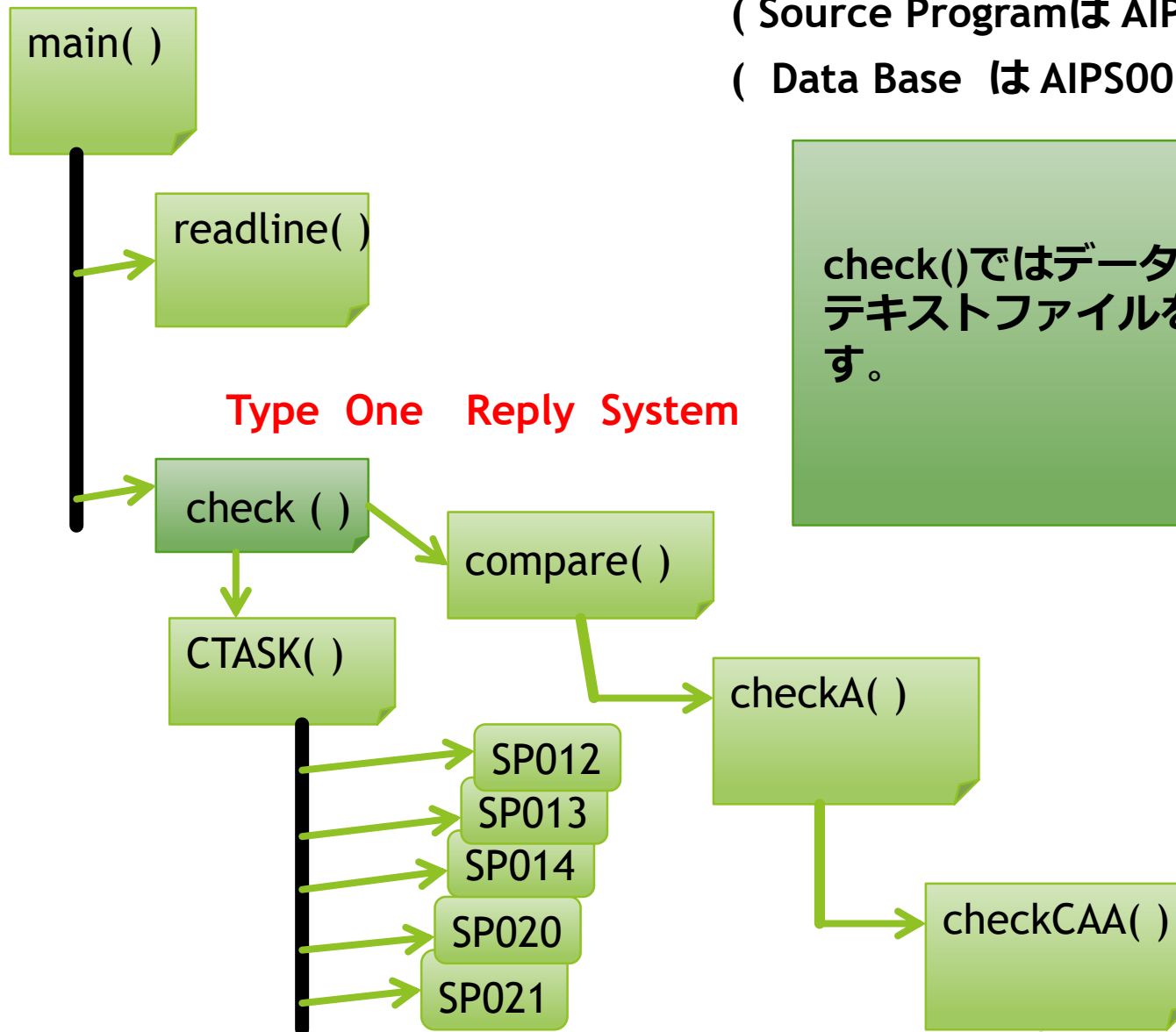
(Data Base は AIPS001DB.txt に格納)



Type One 会話システムProgramの構造

(Source Programは AIPS001.txt に格納)

(Data Base は AIPS001DB.txt に格納)

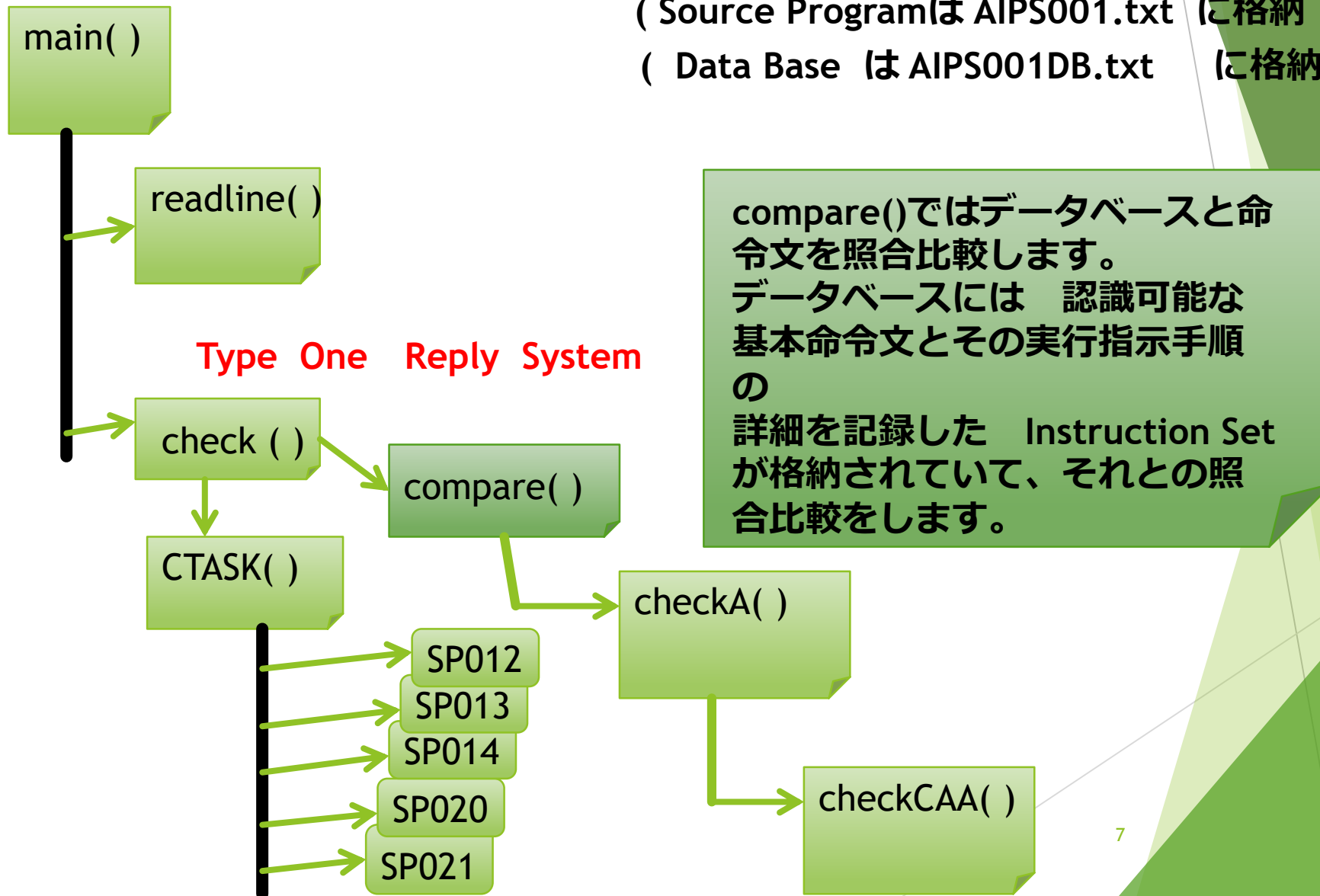


check()ではデータベースとなる、
テキストファイルを読み取りま
す。

Type One 会話システムProgramの構造

(Source Programは AIPS001.txt に格納)

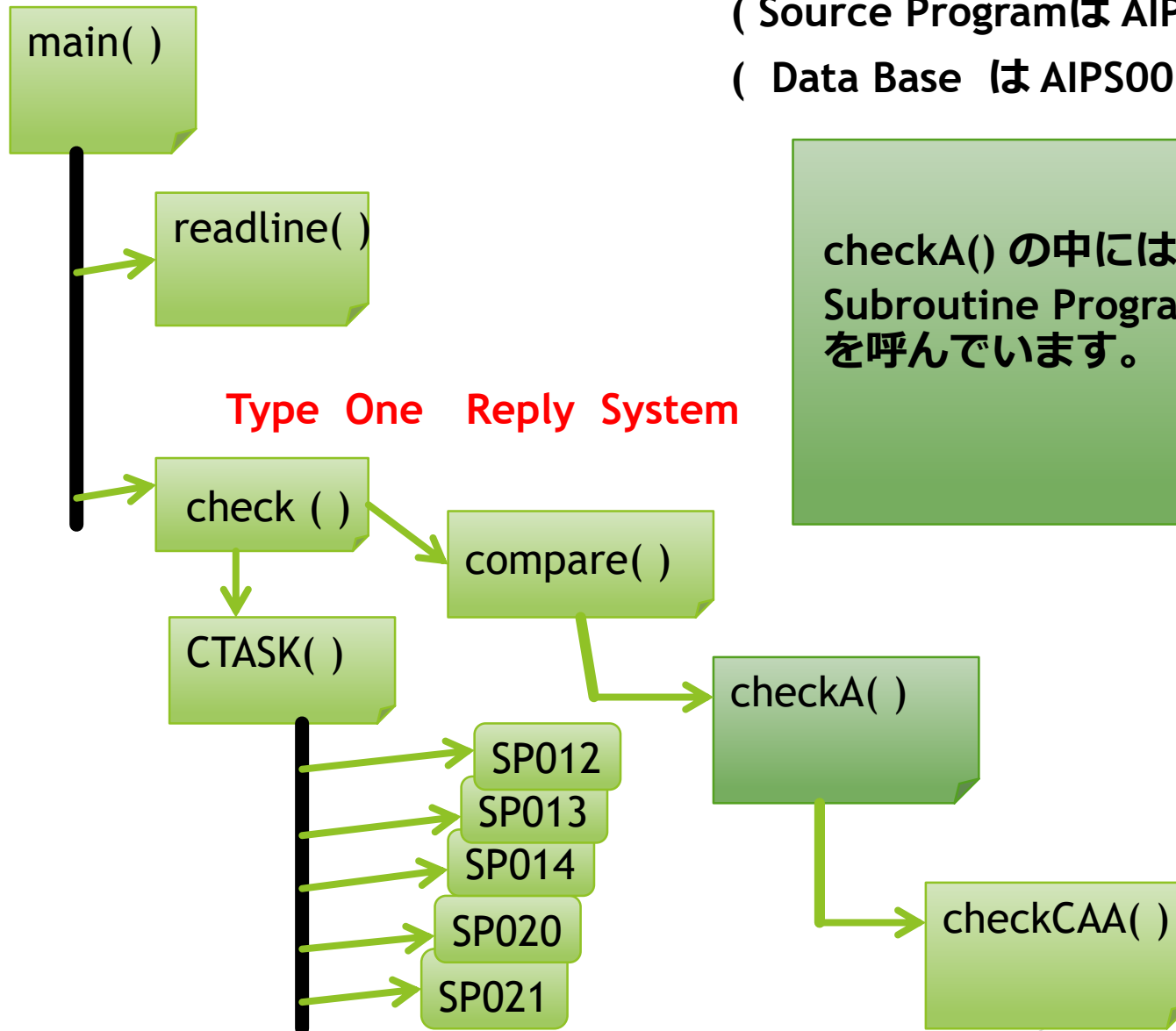
(Data Base は AIPS001DB.txt に格納)



Type One 会話システムProgramの構造

(Source Programは AIPS001.txt に格納)

(Data Base は AIPS001DB.txt に格納)



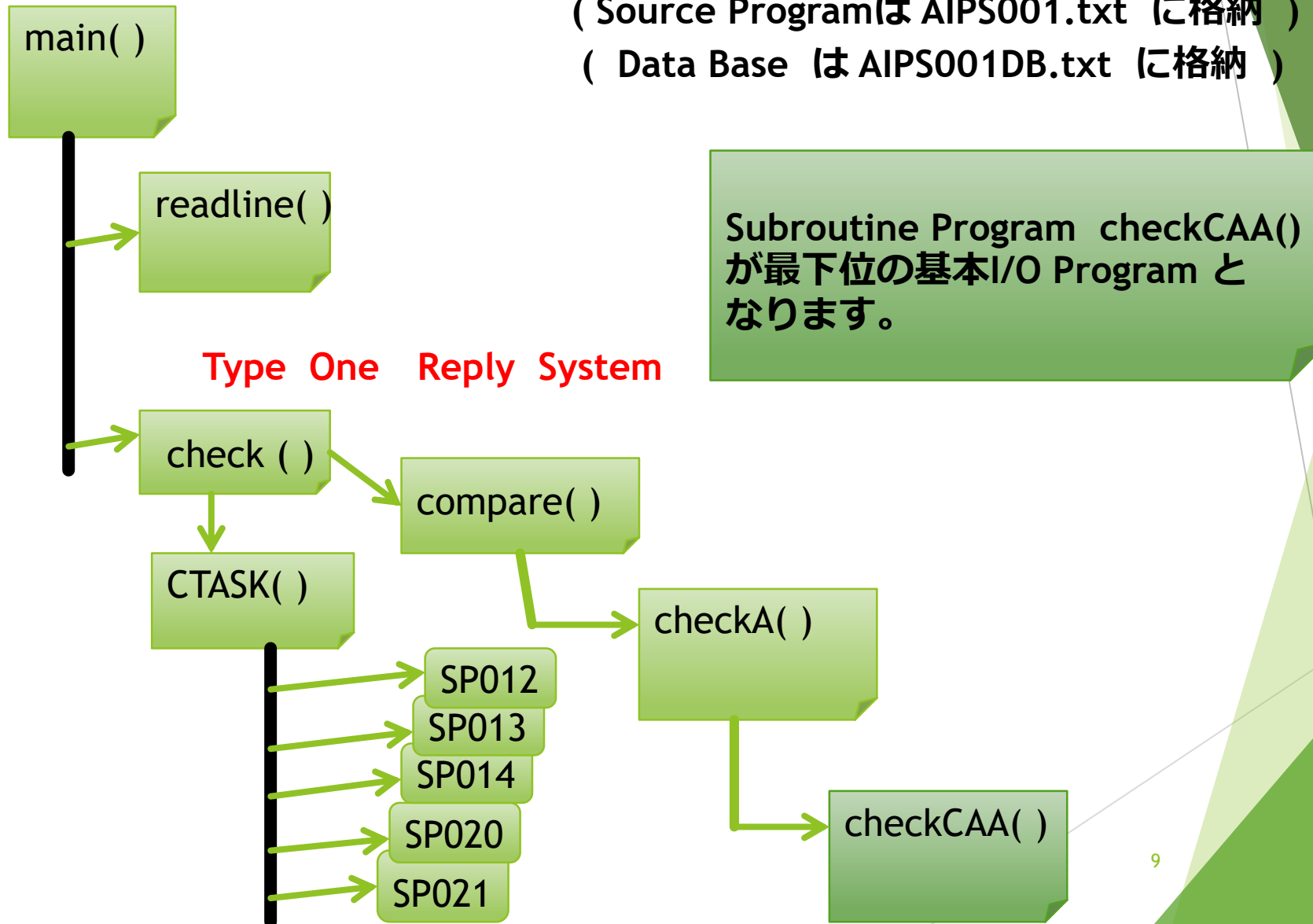
Type One Reply System

checkA()の中にはさらに
Subroutine Program checkCAA()
を呼んでいます。

Type One 会話システムProgramの構造

(Source Programは AIPS001.txt に格納)

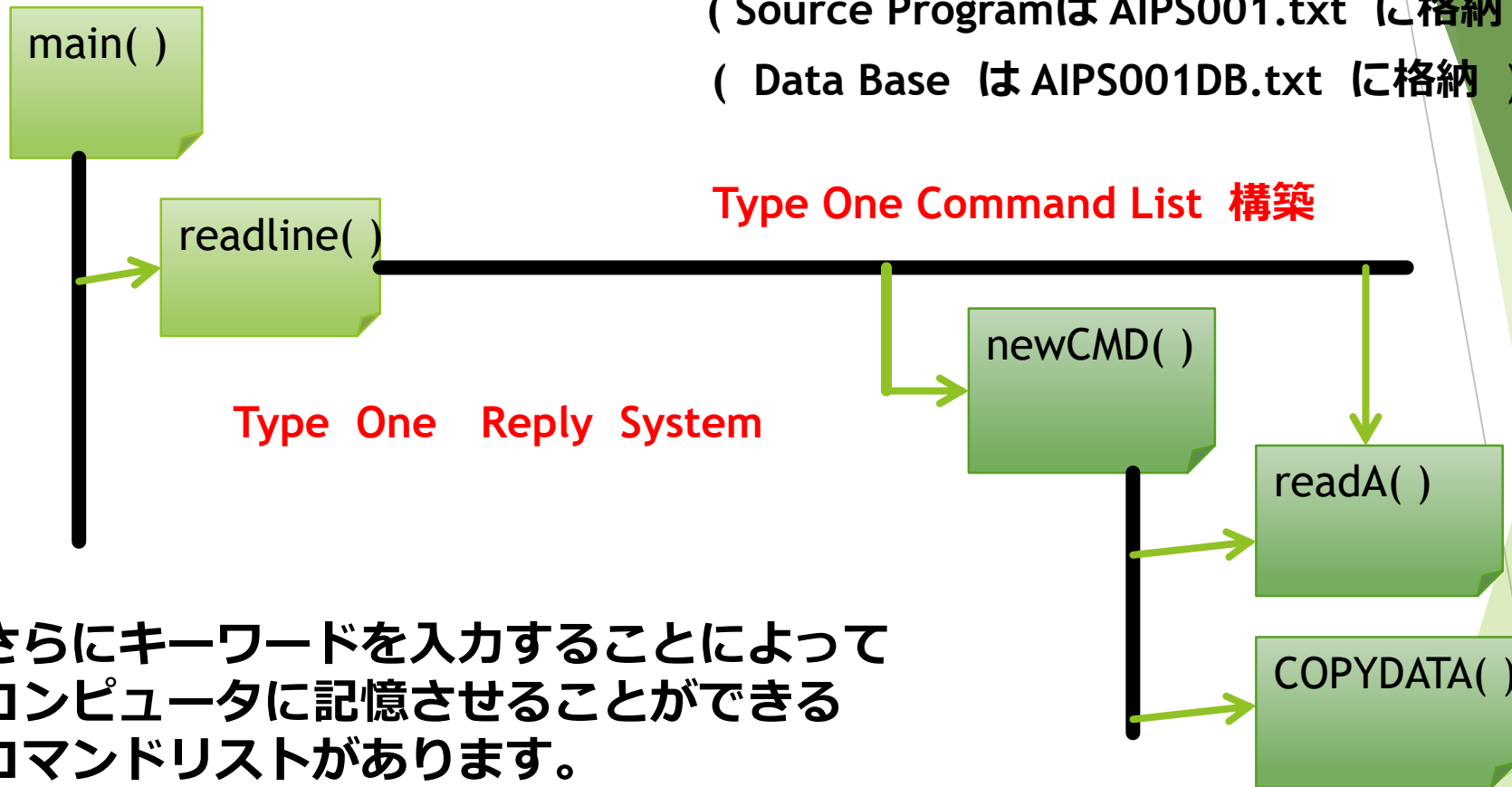
(Data Base は AIPS001DB.txt に格納)



Type One 会話システムProgramの構造

(Source Programは AIPS001.txt に格納)

(Data Base は AIPS001DB.txt に格納)



さらにキーワードを入力することによって
コンピュータに記憶させることができる
コマンドリストがあります。

Type One 会話システムのコマンドリスト

最初、データベース
(AIPsDB001.txt)
には、なにもありません。

人間の赤ちゃんと同じで
記憶しているものは皆無です。

そこで最初は何にをInput しても

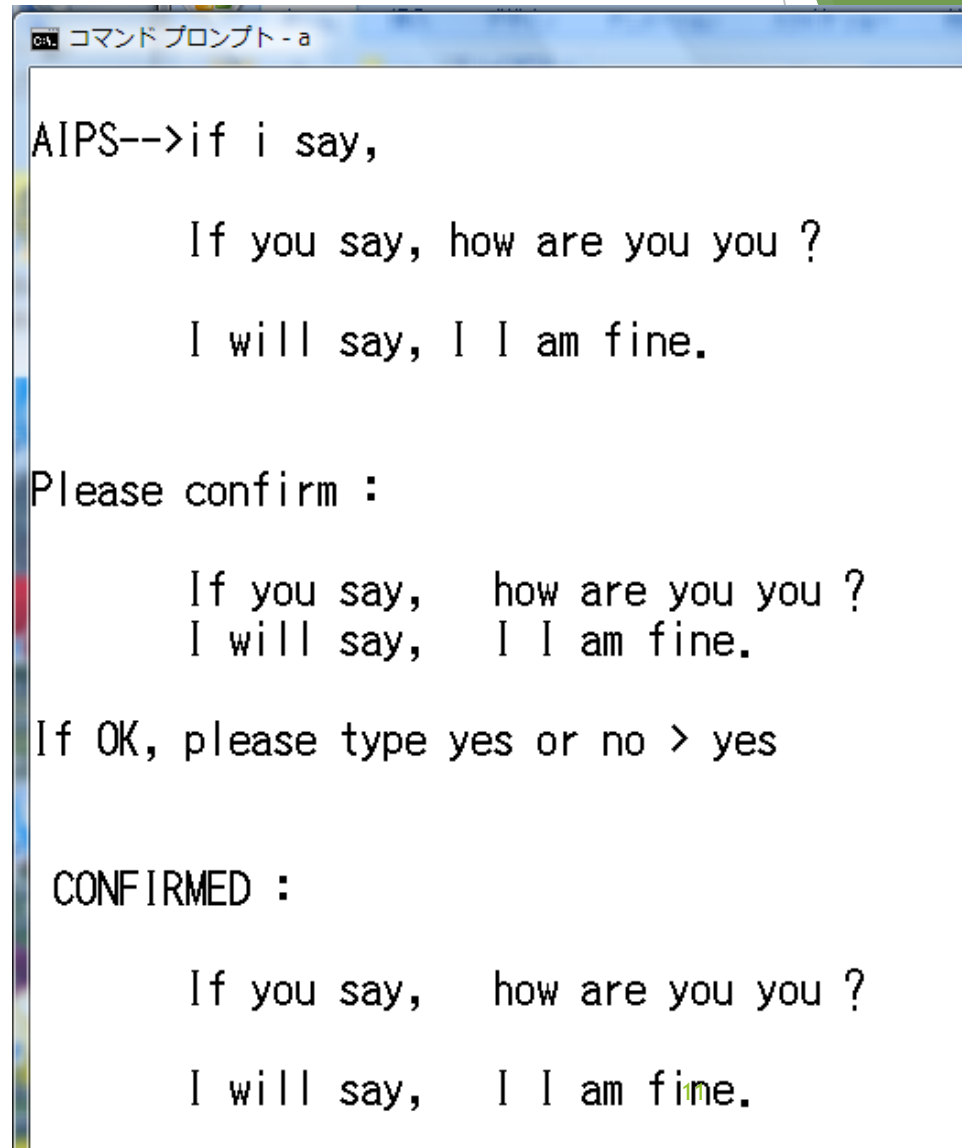
“I don't understand ...”

と システムは返事します。

ただし、**Key Word** “if i say, ” が
入力されますと、システムは
その時のみ

“ If you say, ← 質問
“ I will say, ← 解答

の形式で次の入力指示を待ちます。



```
04. コマンドプロンプト - a
AIPS-->if i say,
    If you say, how are you you ?
    I will say, I I am fine.
Please confirm :
    If you say, how are you you ?
    I will say, I I am fine.
If OK, please type yes or no > yes
CONFIRMED :
    If you say, how are you you ?
    I will say, I I am fine.
```

Type One 会話システムのコマンドリスト

その結果 Data Base (AIPS001DB.txt)
にCommand が1個登録されます。

NCMD = 1

COMMAND[1] = 001

REQUEST[1] = 4

REQ[1,1] = how

REQ[1,2] = are

REQ[1,3] = you

REQ[1,4] = you

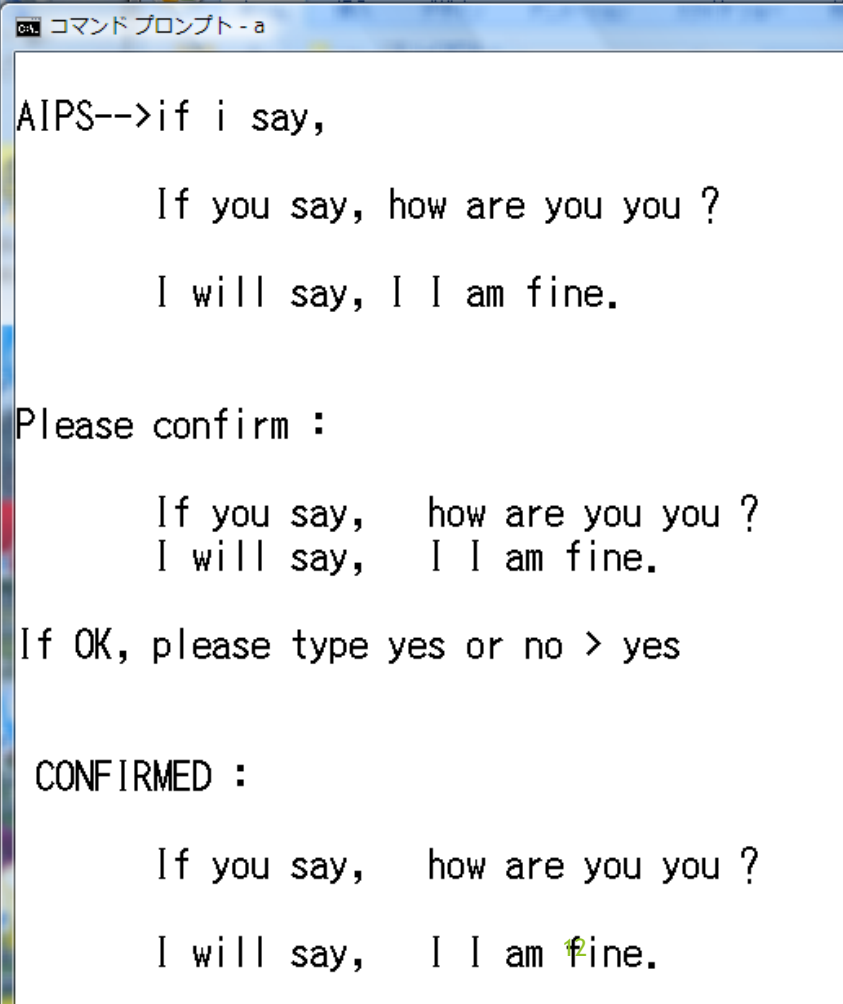
REPLY[1] = 4

REP[1,1] = I

REP[1,2] = I

REP[1,3] = am

REP[1,4] = fine



```
コマンドプロンプト - a
AIPS-->if i say,
    If you say, how are you you ?
    I will say, I I am fine.

Please confirm :
    If you say, how are you you ?
    I will say, I I am fine.

If OK, please type yes or no > yes

CONFIRMED :
    If you say, how are you you ?
    I will say, I I am fine.
```

Type One 会話システムのコマンドリスト

このように “if i say” を key word とする command 手続きを経て Data Base(AIPS001DB.txt) には下記のような内容も登録が可能です。実際に実行すると AIPS システムは 指示どおりに返答します。

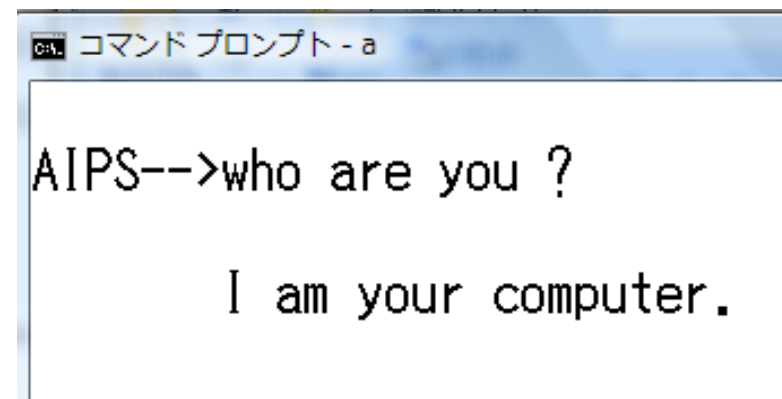
COMMAND[2] = 002

REQUEST[2] = 3

REQ[2,1] = who
REQ[2,2] = are
REQ[2,3] = you

REPLY[2] = 4

REP[2,1] = I
REP[2,2] = am
REP[2,3] = your
REP[2,4] = computer



```
コマンドプロンプト - a
AIPS-->who are you ?
I am your computer.
```

If I sayをキーワードとするコマンドを使用してRequest2とReply2でこのように出力されます。

Type One 会話システムのコマンドリスト

同様に “if i say” を key word とする command 手続きを経て Data Base(AIPS001DB.txt) には下記のような内容も登録が可能です。実際に実行すると AIPS システムは 指示どおりに返答します。

COMMAND[3] = 003

REQUEST[3] = 3

REQ[3,1] = what

REQ[3,2] = are

REQ[3,3] = you

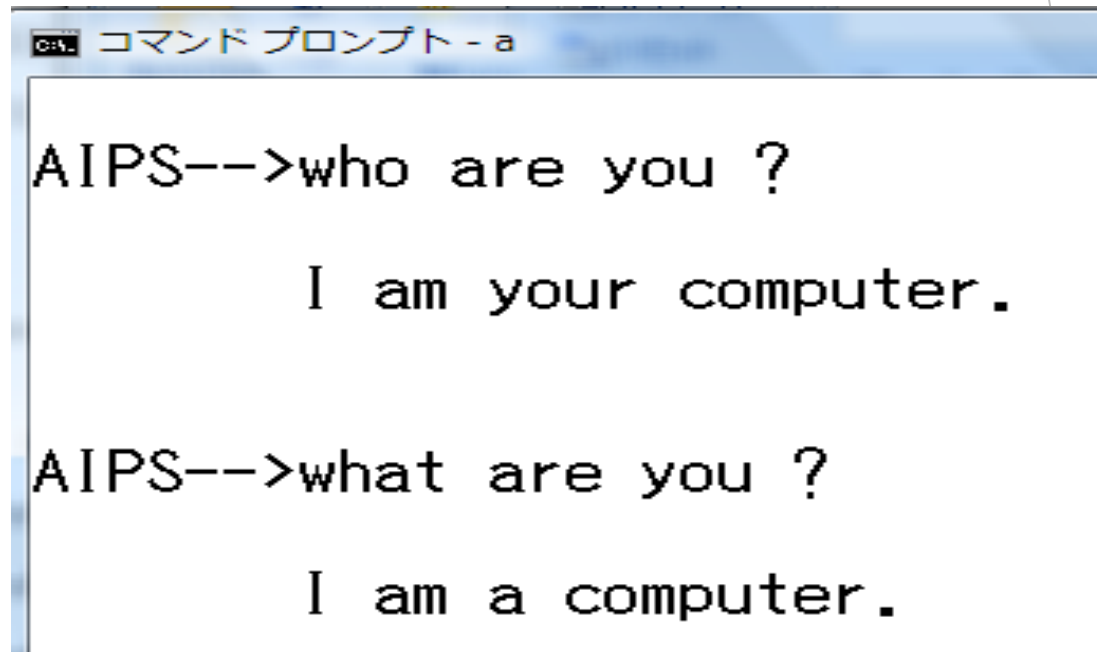
REPLY[3] = 4

REP[3,1] = I

REP[3,2] = am

REP[3,3] = a

REP[3,4] = computer



```
コマンドプロンプト - a

AIPS-->who are you ?

I am your computer.

AIPS-->what are you ?

I am a computer.
```

さらに、Request3とReply3
で似たようなことを記憶させ
ても
記憶させた通りに出力されま
す。

Type One 会話システムの別途Program

また別途用意されているいろいろな 計算Program も連結することが可能です。
たとえば、Data Base (AIPS001DB.txt)の中の21番目の項目(P021)には次のようなdataも格納が可能です。**これは任意のP進法数字をQ進法数字に変換Program (PtoQ)**が登録されている事を示します。

COMMAND(21) = 021P

REQUEST(21) = 1

REQ[21,1] = PtoQ
REPLY(21) = 7

REP[21,1] = This
REP[21,2] = program
REP[21,3] = converts
REP[21,4] = P-system
REP[21,5] = number
REP[21,6] = to
REP[21,7] = Q-system
REP[21,8] = number

```
コマンドプロンプト - a
AIPS-->PtoQ ?

P021 = **** P 進法数を Q進法数に変換します。****

INPUT NUMBER SYSTEM P = 5
                        P = 5 = (00101)

OUTPUT NUMBER SYSTEM Q = 9
                        Q = 9 = (01001)

Give p=12043 ?

P=12043 <-- Number System P = 5

D=5267 <-- Decimal Number

B=1010010010011 <-- Binary Number

Q=7202 <-- Number System Q = 9

AIPS-->
```

TYPE ONEでは、

「A is B 」 「B is C」と一問一答でした。

しかし、AとCが同じであればAを命令してもCが実行されるよう
LINK/NESTING機能 (Type2)が必要だと考えました。

TYPE ONE
What is A ?
A is B .

TYPE ONE
What is
B ?
B is C.

TYPE TWO
What is A ? A is B .
What is B ? B is C.
Is A C ? Yes, A is C.

LINK/NESTING機能

LINK/NESTING機能

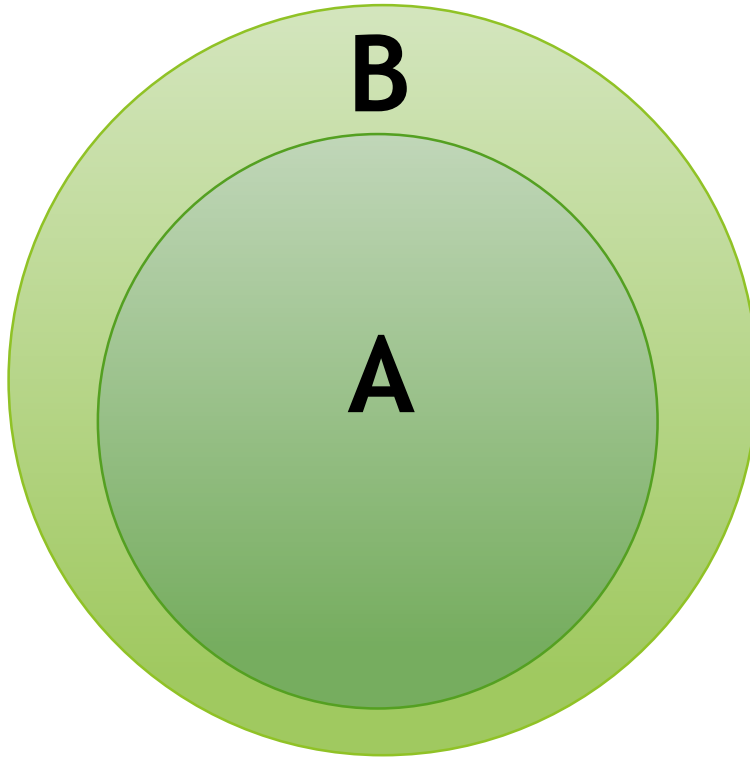


図 1

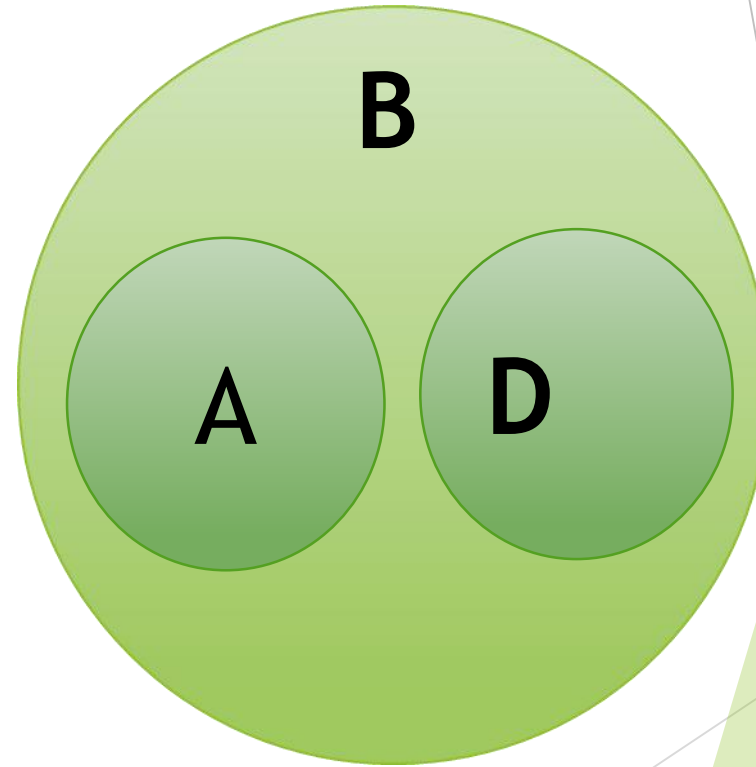


図 2

様々な集合体

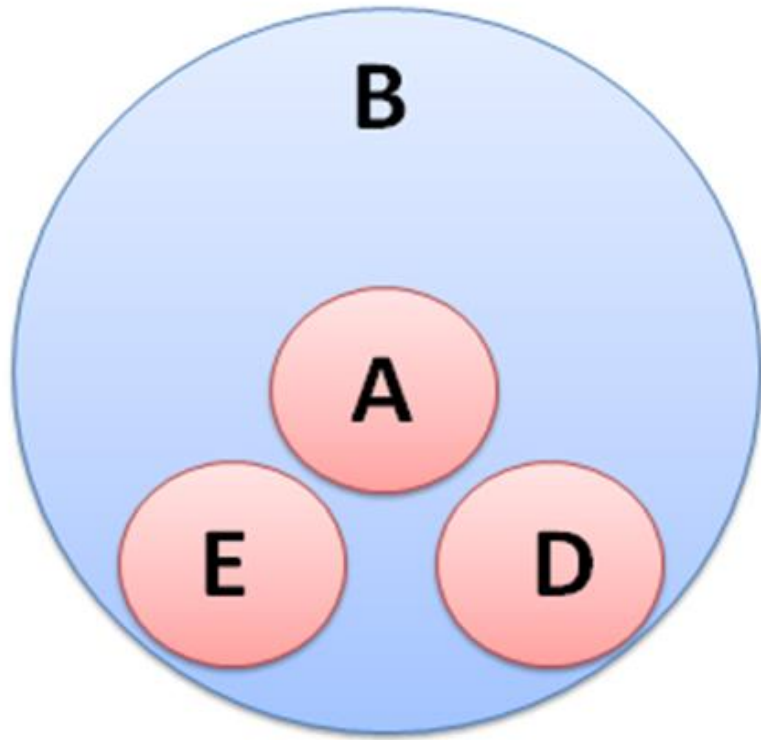


図3

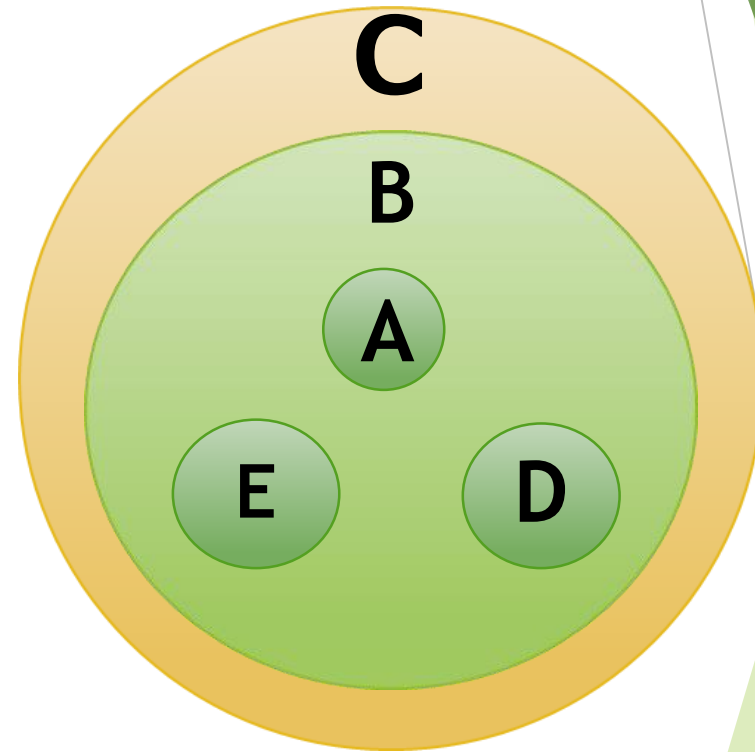


図4

图 1

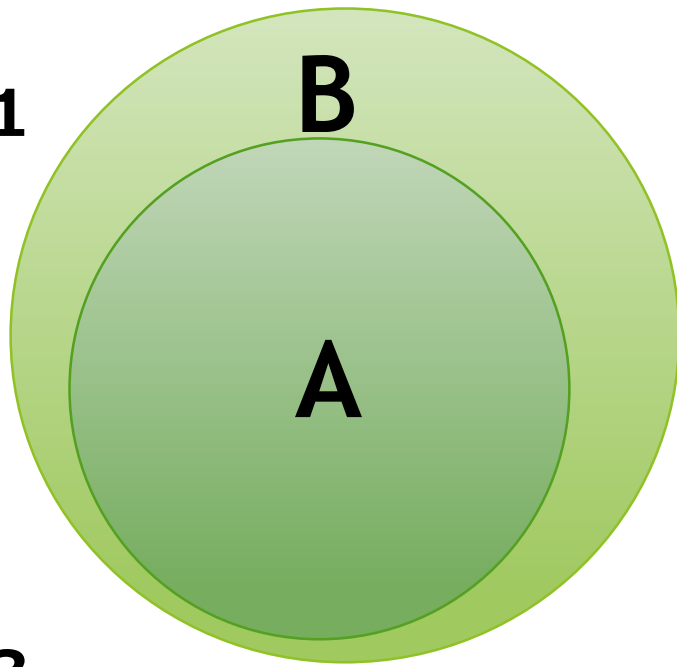


图 2

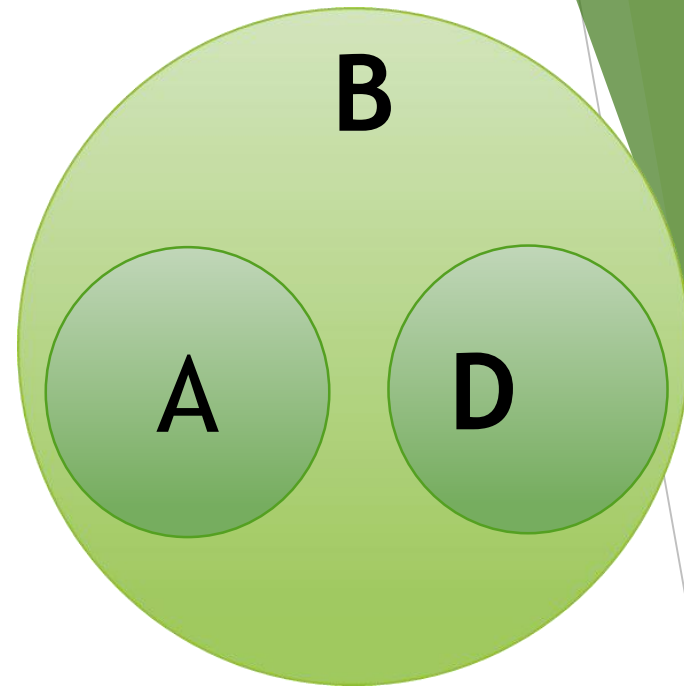


图 3

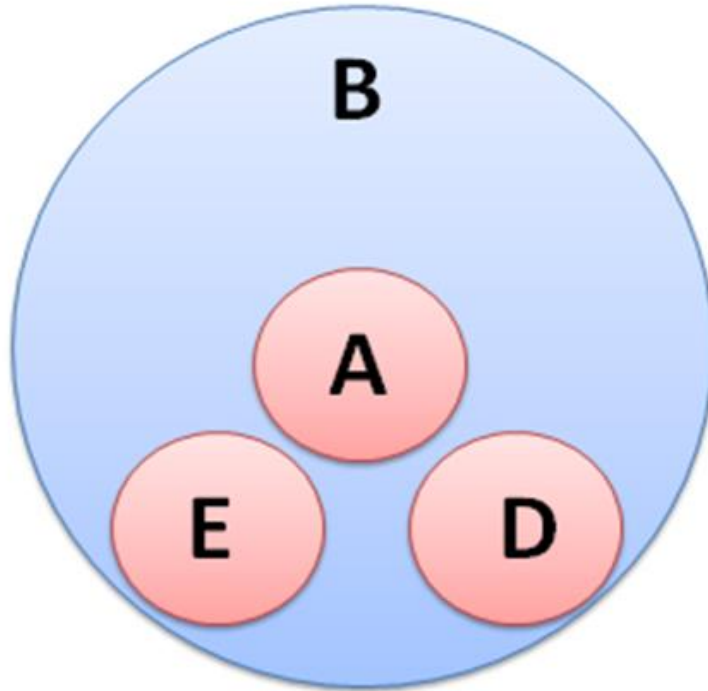
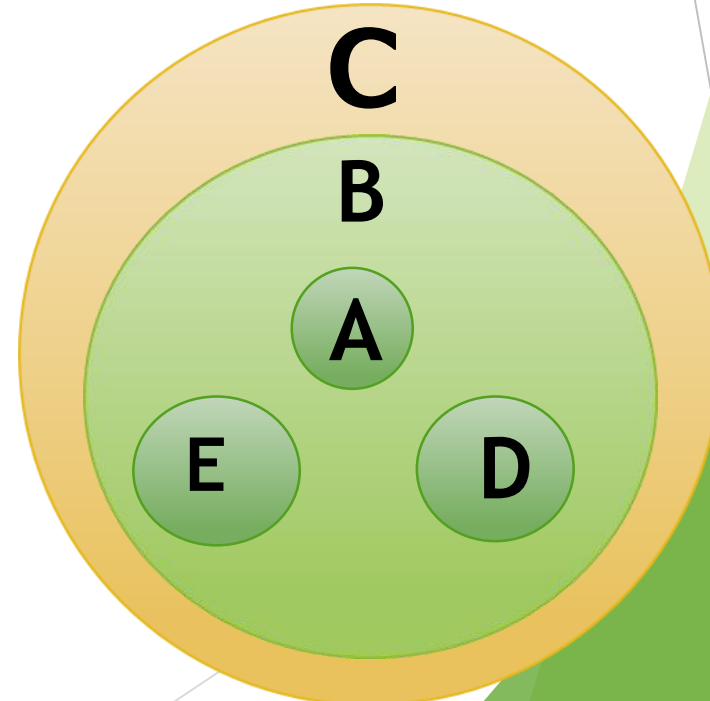


图 4



TYPE ONEでは、

「A is B 」 「B is C」と一問一答でした。

しかし、AとCが同じであればAを命令してもCが実行されるよう
LINK/NESTING機能 (Type2)が必要だと考えました。

TYPE ONE
What is A ?
A is B .

TYPE ONE
What is
B ?
B is C.

TYPE TWO
What is A ? A is B .
What is B ? B is C.
Is A C ? Yes, A is C.

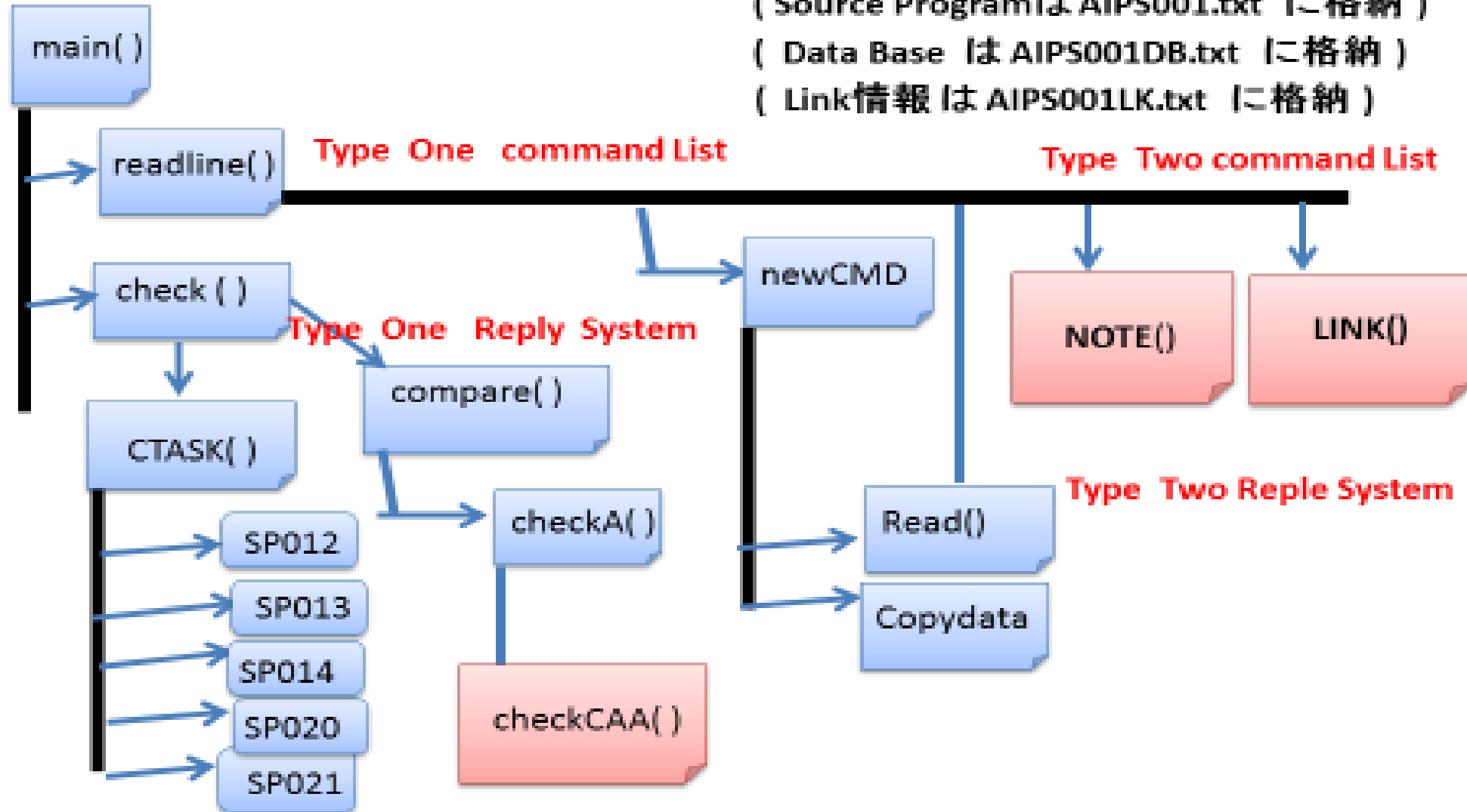
LINK/NESTING機能

Type Two LINK/NESTING 会話システム

(Source ProgramはAIPS001.txt に格納)

(Data Base はAIPS001DB.txt に格納)

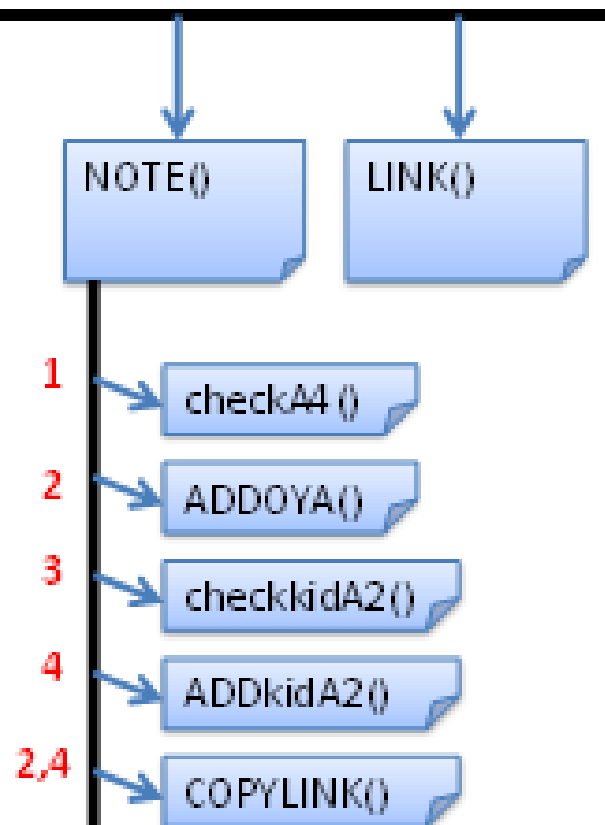
(Link情報はAIPS001LK.txt に格納)



Type Two LINK/NESTING を組み込んだ会話システムの構築

Type Two Reply System

Type Two Command List 構築



(Source Program は AIPS001.txt に格納)

(Data Base は AIPS001DB.txt に格納)

(Link 情報は AIPS001LK.txt に格納)

int NOTE(void)

- (1) A4 が OYA-LIST にあるかがす。
- (2) A4 が OYA-LIST にない場合
ADD OYA() と COPYLINK() を実行
- (3) A4 が OYA-LIST にある場合
A4 = OYA ofiset となり、
checkkidA2() を実行
A2 が KID-LIST にあると
“I Knew ...” と返事し 終了。
- (4) A2 が KID-LIST にないと
ADDkidA2() と COPYLINK() を実行。

Type Two LINK/NESTINGを組み込んだ会話システムの構築

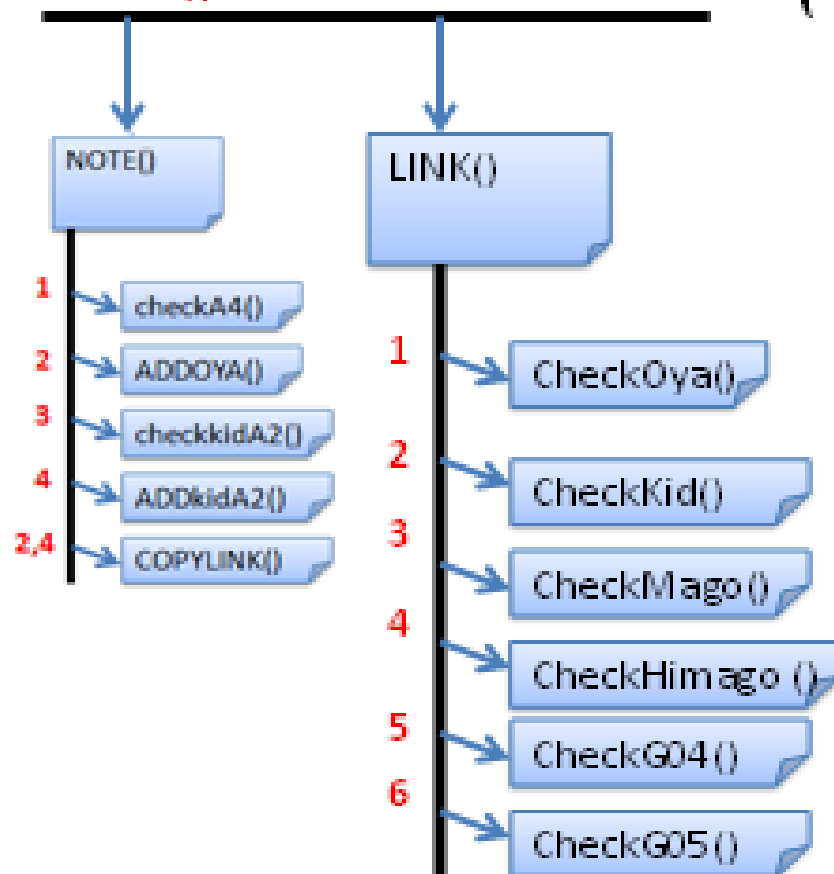
Type Two Reply System

(Source Program は AIPS001.txt に格納)

(Data Base は AIPS001DB.txt に格納)

(Link 情報は AIPS001LK.txt に格納)

Type Two Command List 構築



int LINK(void)

- (1) A4 が OYA-LIST にあるかがず。
A4 が OYA-LIST がない場合
“I don't know A4 ...” と返答。
- (2) A4 が OYA-LIST にある場合
A4 = OYA ofiset となり
CheckKid() を実行
A3 が KID-LIST にあると
“Yes A3 is A4 . ” と返事。
- (3) A3 が A4 の Mago LIST にいるか
どうかをcheckします。いれば
“Yes A3 is A4 . ” と返事。
- (4) A3 が A4 の Himago LIST にいるか
どうかをcheckします。いれば
“Yes A3 is A4 . ” と返事。
- (5) 以下 同様...

LINK/NESTING機能について説明します。

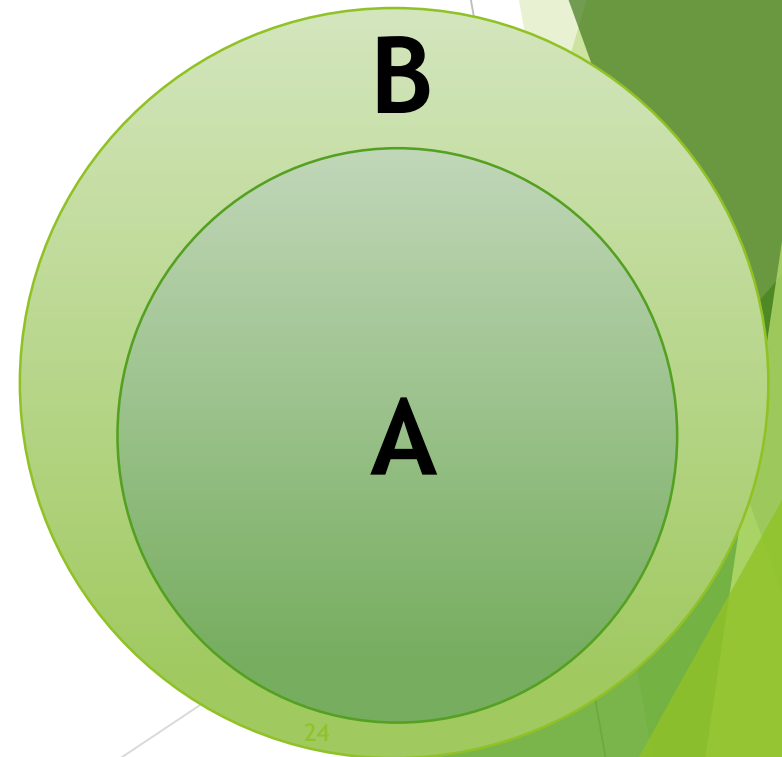
まず“**NOTE A is B.**” と入力した場合です。
Bの中にAが含まれており、Bが親でAが子となっています。

親の数 → nset = 1

親 → set name[1] = 'B'

子の数 → set element[1] = 1

子 → set element name[1,1] = 'A'



さらに“**NOTE D is B.**”と入力した場合です。

Bの子がAとDになります。

システムData Base には 次のような変数値が設定されます。

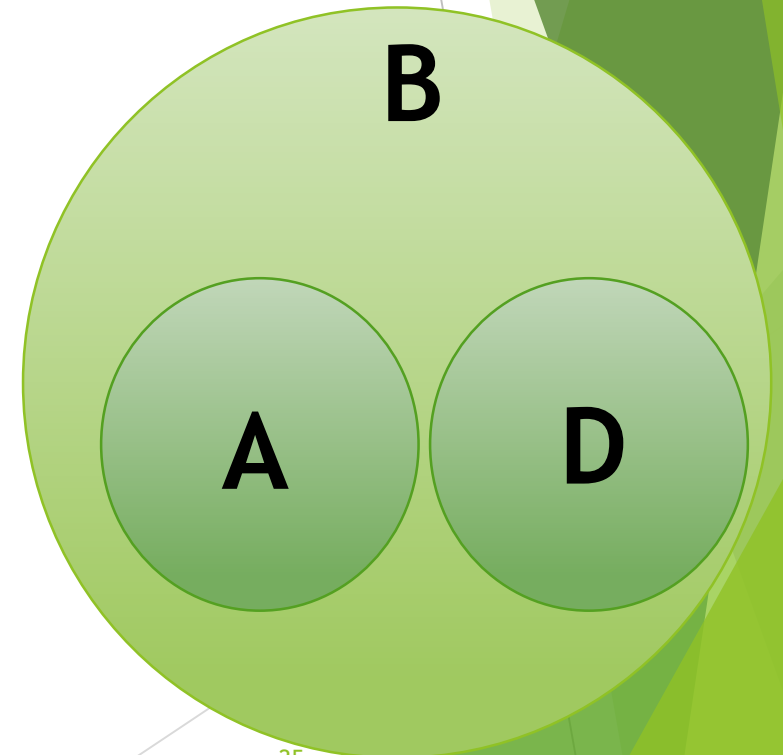
親の数 → nset = 1

親 → setname[1]='B'

子の数 → setelement[1]=2

子 → setelementname[1,1]= 'A'

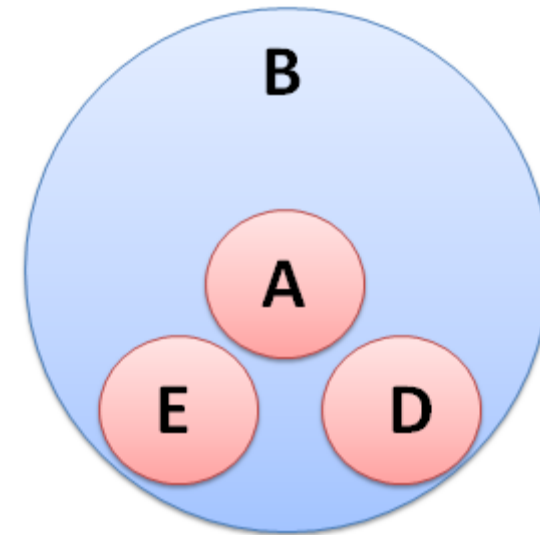
子 → setelementname[1,2]= 'D'



さらに“NOTE E is B.” と入力した場合です。
システムData Base には 次のような変数値が設定されます。

親の数 → nset = 1

親 → setname[1]='B'
子 → setelement[1]=3
子 → setelementname[1,1]= 'A'
子 → setelementname[1,2]= 'D'
子 → setelementname[1,3]= 'E'



システムData Baseに,これらのデータが格納されると、
質問“ Is A B ?”に対して「Yes, A is B」 と返事します。

そこで“ **NOTE B is C .**” と入力します。
そうすることで親であったBがCに含まれ、
CがBの親になります。

システムData Base には 次のような変数値が設定されます。

親の数 → nset = 2

親 → setname[1]='B'

子の数 → setelement[1]=3

子 → setelementname[1,1]= 'A'

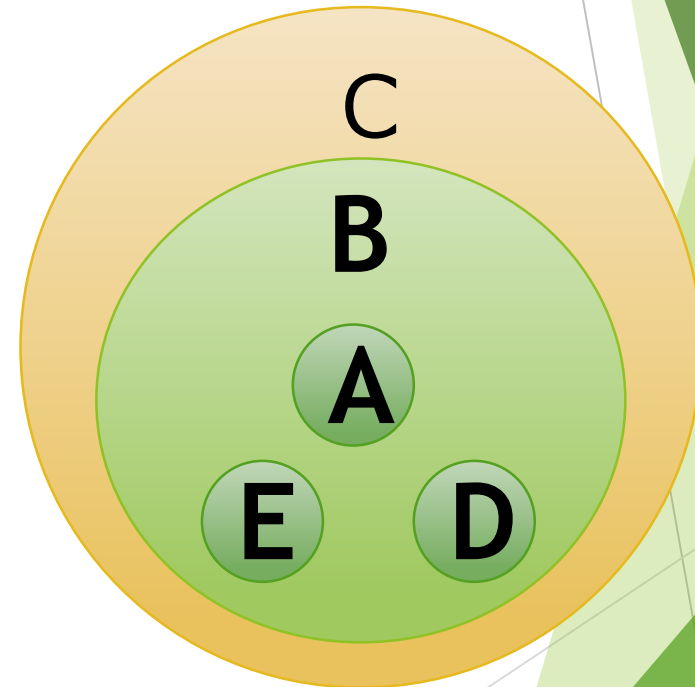
子 → setelementname[1,2]= 'D'

子 → setelementname[1,3]= 'E'

setname[2]='C'

setelement[2]=1

setelementname[2,1]= 'B'



さらに“**NOTE F is C.**”と入力した場合、
Cの中にFが含まれることになり、このようにどんどん増やす
ことができます。

システムData Base には 次のような変数値が設定されます。

親の数 → nset = 2

親 → setname[1]='B'

子の数 → setelement[1]=3

子 → setelementname[1,1]='A'

子 → setelementname[1,2]='D'

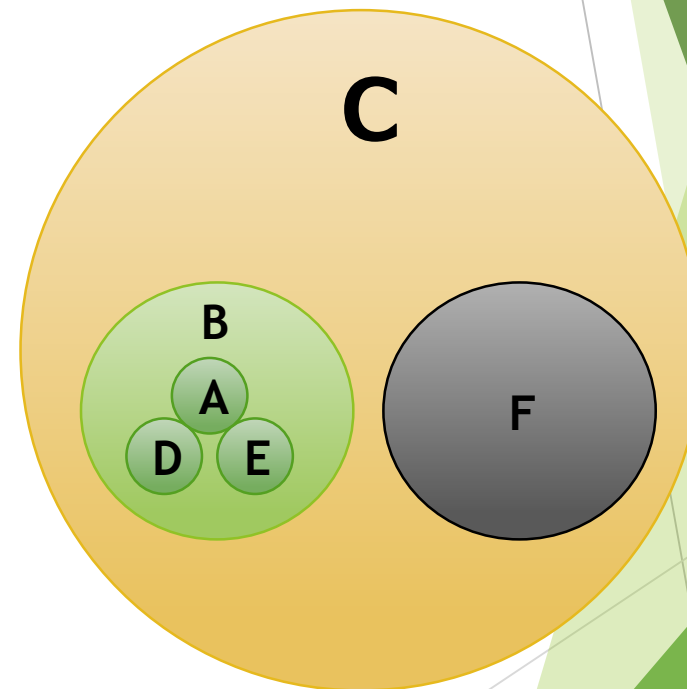
子 → setelementname[1,3]='E'

setname[2]='C'

setelement[2]=2

setelementname[2,1]='B'

setelementname[2,2]='F'



(2) しかし、Aが格納されている場所を見つけますと
set element name Aを抽出します。

nset = 2

setname[1]='B'

setelement[1]=3

→ setelementname[1,1]='A' →

setelementname[1,2]='D'

setelementname[1,3]='E'

setname[2]='C'

setelement[2]=2

setelementname[2,1]='B'

setelementname[2,2]='F'

setelementname[1,1]='A'

相関抽出情報

(3)さらに、その親である set name Bを抽出します。

nset = 2

→ setname[1]='B'
setelement[1]=3
setelementname[1,1]='A'
setelementname[1,2]='D'
setelementname[1,3]='E'

setname[2]='C'
setelement[2]=2
setelementname[2,1]='B'
setelementname[2,2]='F'

setelementname[1,1]='A'
setname[1]='B'

相関抽出情報

(4)今度はset element name[,]内の 'B'が格納されている
変数をさがします。
なければ "I do not know. I have no data."と返事します。

```
nset = 2
```

```
setname[1]='B'
```

```
setelement[1]=3
```

```
setelementname[1,1]= 'A'
```

```
setelementname[1,2]= 'D'
```

```
setelementname[1,3]= 'E'
```

```
setname[2]='C'
```

```
setelement[2]=2
```

```
setelementname[2,1]= 'B'
```

```
setelementname[2,2]= 'F'
```

```
setelementname[1,1]= 'A'  
setname[1]='B'
```

相関抽出情報

(5) しかし、Bが格納されている場所を見つけますと
set element name Bを抽出します。

nset = 2

setname[1]='B'

setelement[1]=3

setelementname[1,1]='A'

setelementname[1,2]='D'

setelementname[1,3]='E'

setname[2]='C'

setelement[2]=2

→ setelementname[2,1]='B'

setelementname[2,2]='F'

setelementname[1,1]='A'
setname[1]='B'

setelementname[2,1]='B'

→ 相関抽出情報

(6) その親であるsetname[2]='C'を抽出します。

nset = 2

setname[1]='B'

setelement[1]=3

setelementname[1,1]= 'A'

setelementname[1,2]= 'D'

setelementname[1,3]= 'E'

→ setname[2]='C'

setelement[2]=2

setelementname[2,1]= 'B'

setelementname[2,2]= 'F'

setelementname[1,1]= 'A'
setname[1]='B'

setelementname[2,1]= 'B'
setname[2]='C'

相関抽出情報

nset = 2

setname[1]='B'

setelement[1]=3

setelementname[1,1]= 'A'

setelementname[1,2]= 'D'

setelementname[1,3]= 'E'

setname[2]='C'

setelement[2]=2

setelementname[2,1]= 'B'

setelementname[2,2]= 'F'

これで4つの関係を見つけることができました。

setelementname[1,1]= 'A'
setname[1]='B'

setelementname[2,1]= 'B'
setname[2]='C'

相関抽出情報

これで 'A'の親が 'B'であり、'B'の親が 'C'であることが判明します。
したがって、会話システムは、'C'の集合に'A'を含むと判断します。
そこで、 " Is A C ? " という質問が入ると会話システムは、

" Yes, A is C since A is B and B is C. "

nset = 2

と答えることができます。

setname[1]='B'
setelement[1]=3
setelementname[1,1]='A'
setelementname[1,2]='D'
setelementname[1,3]='E'

setname[2]='C'
setelement[2]=2
setelementname[2,1]='B'
setelementname[2,2]='F'

setelementname[1,1]='A'
setname[1]='B'

setelementname[2,1]='B'
setname[2]='C'

相関抽出情報

このLINK/NESTINGのAlgorithmは 仕事の細分 組織化の定義にも応用できると考えます。work(B)という作業にwork (A)という作業が含まれ、work (C)という作業にwork (B)の作業が含まれる時、work (C)の作業を実行するときはかならずwork (B)の作業を実行する時にwork(A)の作業が実行されることになります。

nset = 2

setname[1]='B'

setelement[1]=3

setelementname[1,1]= 'A'

setelementname[1,2]= 'D'

setelementname[1,3]= 'E'

setname[2]='C'

setelement[2]=2

setelementname[2,1]= 'B'

setelementname[2,2]= 'F'

setelementname[1,1]= 'A'
setname[1]='B'

setelementname[2,1]= 'B'
setname[2]='C'

相関抽出情報

Type Two 会話システム Program の実行例

```
C:\Users\AIPS02\Desktop\卒研TOP20140204\2013最終報告\51031045 栗山侑己\AIPS会話3

***** HelloAIPS *****

Hello, how are you ?

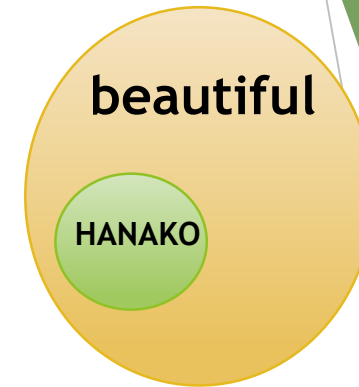
*****

AIPS-->NOTE HANAKO is beautiful .

I did not know HANAKO is beautiful ....

if so, please type yes --> yes

Now I know HANAKO is beautiful ....
```



「HANAKO is beautiful.」と
データベースに記憶させています。

Type Two 会話システム Program の実行例

```
C:\Users\AIPS02\Desktop\卒研TOP20140204\2013最終報告\51031045 栗山南己\AIPS会話:

***** HelloAIPS *****

      Hello, how are you ?

*****

AIPS-->NOTE HANAKO is beautiful .

      I did not know HANAKO is beautiful ....
      if so, please type yes --> yes

      Now I know HANAKO is beautiful ....

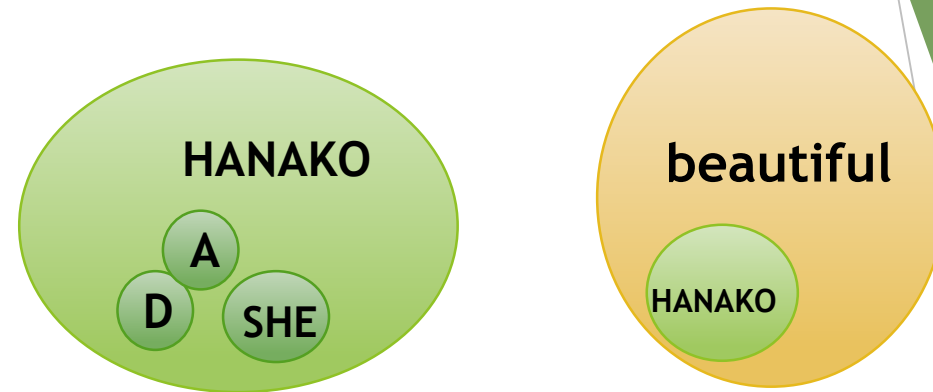
AIPS-->NOTE SHE is HANAKO .

      I did not know SHE is HANAKO ....
      if so, please type yes --> yes

      Now I know SHE is HANAKO ....
```

次に「SHE is HANAKO.」と
記憶させています。

この2つのワードをLINKさせることで
「Is SHE beautiful?」と質問した場合、
右のような返事をします。



```
C:\Users\AIPS02\Desktop\卒研TOP20140204\2013最終報告\51031045 栗山南己\AIPS会話:

AIPS-->LINK is SHE beautiful ?

      Yes, SHE is beautiful ....(MAGO)

      SHE is HANAKO

      HANAKO is beautiful

AIPS-->_
```

Type Two 会話システム Program の実行例

```
C:\Users\AIPS02\Desktop\卒研TOP20140204\2013最終報告\51031045 栗山南己\AIPS会話:

***** HelloAIPS *****

Hello, how are you ?

*****

AIPS-->NOTE HANAKO is beautiful .

I did not know HANAKO is beautiful ....
if so, please type yes --> yes

Now I know HANAKO is beautiful ....

AIPS-->NOTE SHE is HANAKO .

I did not know SHE is HANAKO ....
if so, please type yes --> yes

Now I know SHE is HANAKO ....

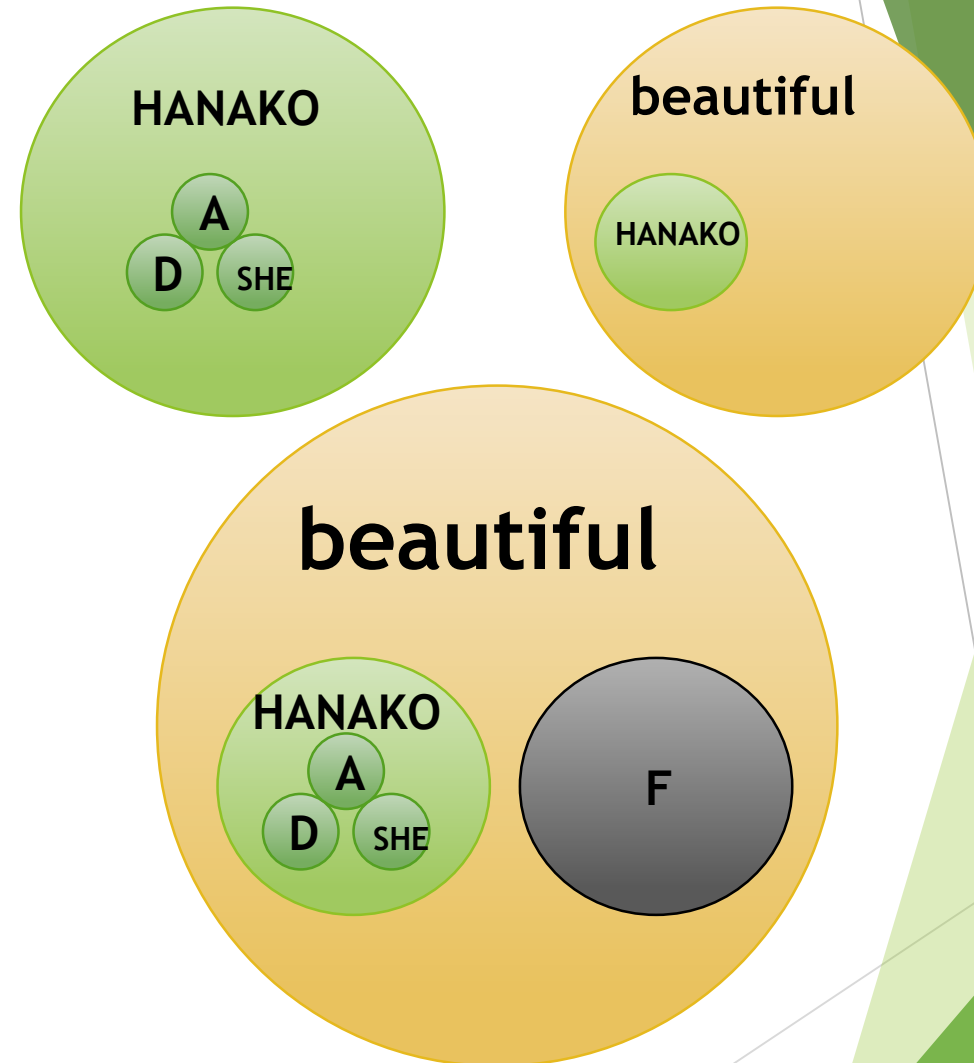
AIPS-->LINK is SHE beautiful ?

Yes, SHE is beautiful ....(MAGO)

SHE is HANAKO

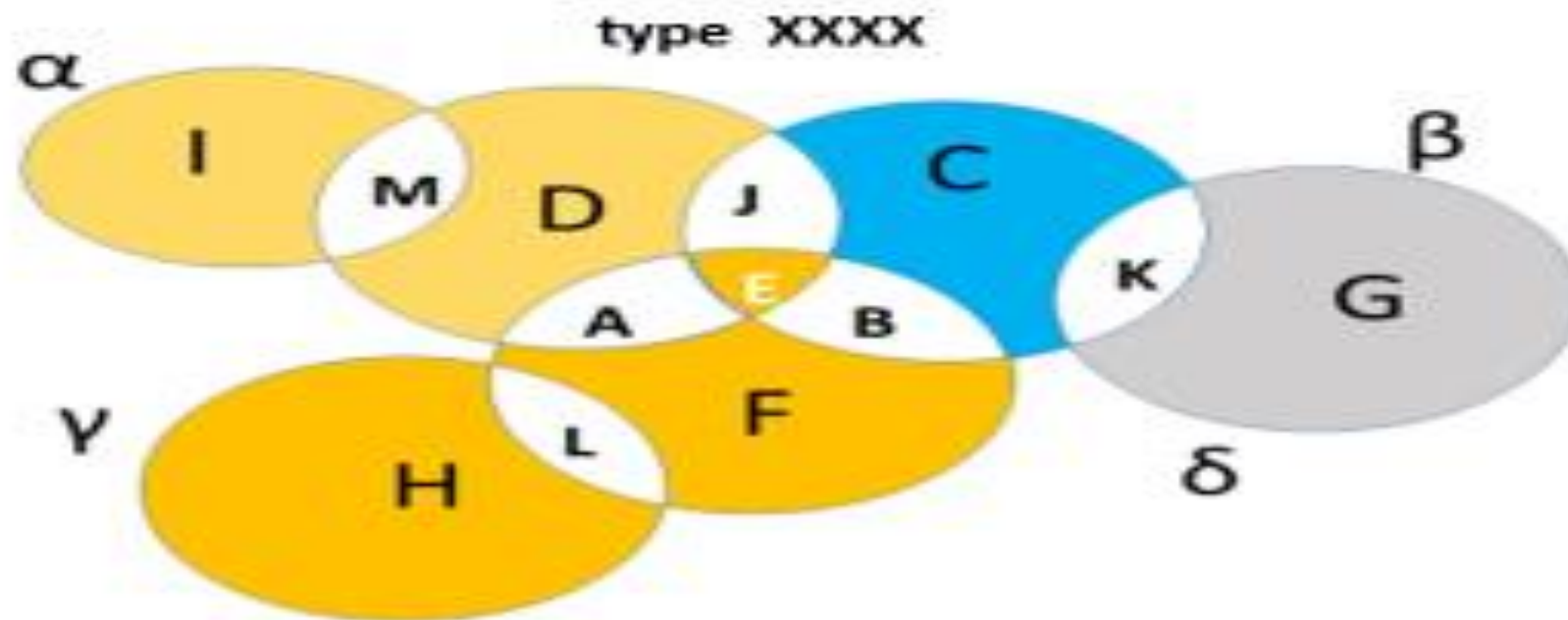
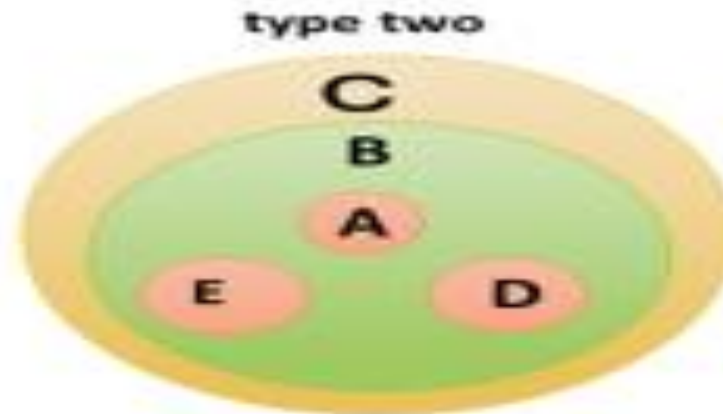
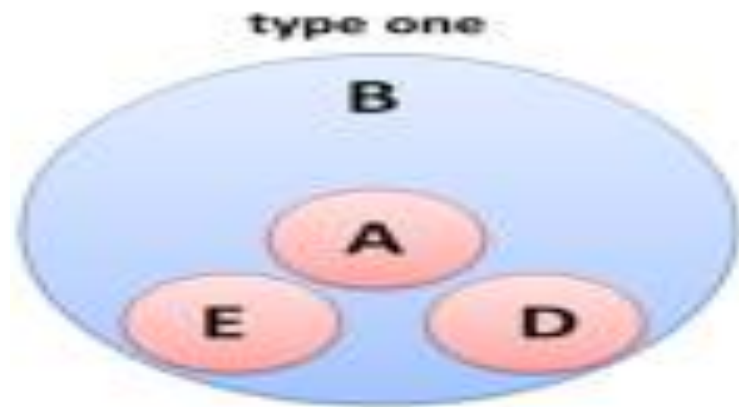
HANAKO is beautiful

AIPS-->_
```



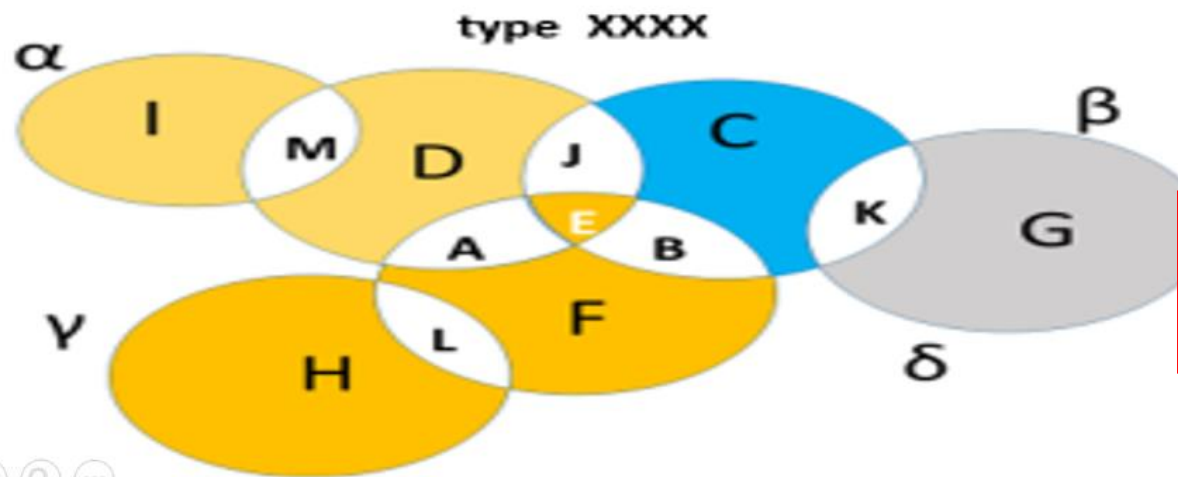
これを繰り返すことで階層状に
さらに増やしていくことができます。

Tree状



データベースを使った会話システム

www.aiplab.com 萩原良昭



END