

Inference in First-Order Logic

Chapter 9

Rules of Inference

$\forall x P(x)$

$\therefore P(c) \text{ if } c \in U$

$P(c) \text{ for an arbitrary } c \in U$

$\therefore \forall x P(x)$

$\exists x P(x)$

$\therefore P(c) \text{ for some element } c \in U$

$P(c) \text{ for some element } c \in U$

$\therefore \exists x P(x)$

از بین بردن سور عمومی : اگر جایی اثبات کرده باشیم به ازای هر x داشته داریم ($p(x)$)
بنابراین در خط جدید می توان (c) را برای یک object از دنیامون میتوان در نظر گرفت

Universal instantiation

اگر یک (c) برای یک p که مشخص نیست با شرطی بر روی c نیست درست باشد
آن گاه می توان در خط جدید سور عمومی اضافه کرد

Universal generalization

از بین بردن سور وجودی : اگر داشته باشیم که یک x ای وجود دارد که $p(x)$ برقرار است
می توانیم این سور را از بین برد و یک اسم برای این x انتخاب کرد مثلاً c که یک چیز مشخص است

Existential instantiation

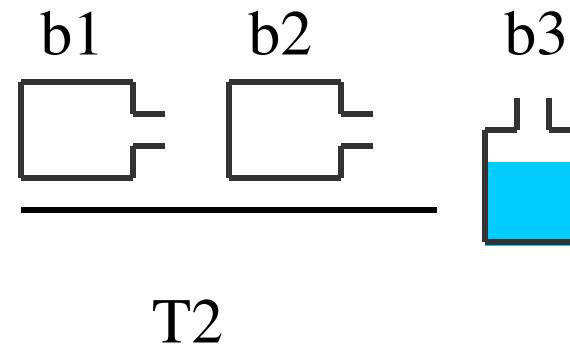
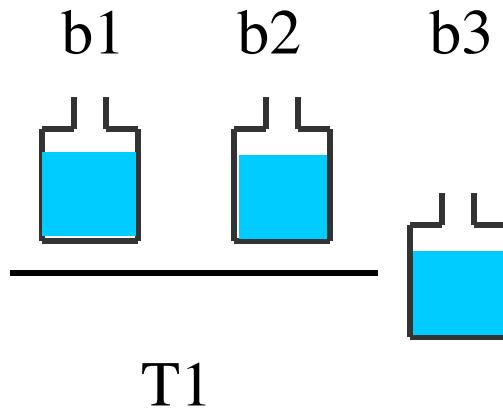
اگر برای تعدادی از c ها از دنیا ثابت کردیم که $p(c)$ برقرار است
می توان سور وجودی را اضافه کرد.

Existential generalization

An example of U.G.

Universal Generalization

در این مثال چون فقط ظرف وجود دارد بنابراین نیازی به تعریف کردن خود ظرف نداریم
یعنی نیازی نیست بگوییم به ازای هر x که x یک ظرف باشد .



1. $\forall x(\text{Ontable}(x, T1) \Rightarrow \text{Upturned}(x, T2))$
2. $\forall x \forall y(\text{Upturned}(x, y) \Rightarrow \text{Empty}(x, y))$

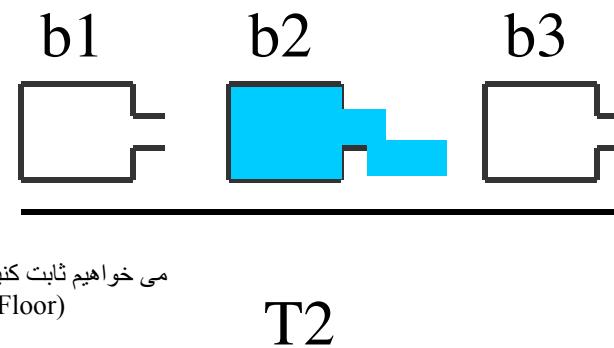
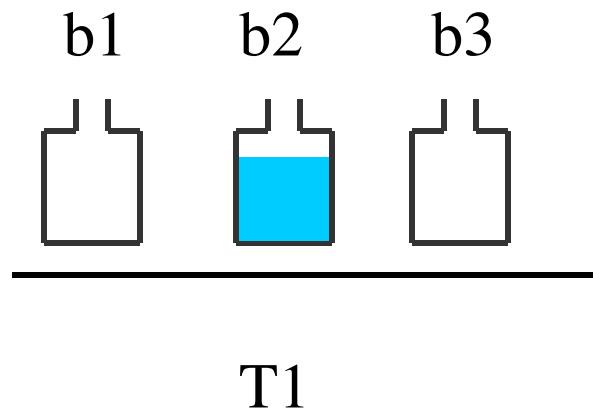
به ازای هر زمان y

An example of U.G.

- | | | |
|----|--|------------------------|
| 3. | Ontable(b2, T1) | Assumption |
| 4. | Ontable(b2, T1) \Rightarrow Upturned(b2, T2) | - \forall , 1 |
| 5. | Upturned (b2, T2) | - \Rightarrow , 3, 4 |
| 6. | Upturned(b2, T2) \Rightarrow Empty(b2, T2) | - \forall , 2 |
| 7. | Empty(b2, T2) | - \Rightarrow , 5, 6 |
| 8. | Ontable(b2, T1) \Rightarrow Empty(b2, T2) | + \Rightarrow , 7 |
| 9. | $\forall x(Ontable(x, T1) \Rightarrow Empty(x, T2))$ | + \forall , 8 |

An example of E.I.

Existential Instantiation



می خواهیم ثابت کنیم زمین خیس است
Wet(Floor)

1. $\forall x(\text{Bottle}(x, T1) \Rightarrow \text{Upturned}(x, T2))$
2. $\forall x \forall y(\text{Upturned}(x, y) \Rightarrow \text{Empty}(x, y))$
3. $\forall x(\text{Full}(x, T1) \& \text{Empty}(x, T2) \Rightarrow \text{Wet}(\text{Floor}))$
4. $\exists x (\text{Bottle}(x, T1) \& \text{Full}(x, T1))$

An example of E.I.

5.	Bottle(b1, T1) & Full(b1, T1))	- EI assumption
6.	Bottle(b1, T1)	- & , 5
7.	Full(b1, T1)	- & , 5
8.	Upturned (b1, T2)	می توانیم (Wet(Floor) را به بدنے اصلی اضافه کنیم چون در عبارت آن خبری از b1 نیست
9.	Empty(b1, T2)	بنابراین در هر صورت درست است پس می توان آن را به بدنے اصلی اضافه کرد
10.	Full(b1, T1) & Empty (b1, T2)	- \forall , 1 , - \Rightarrow , 6
11.	Wet(Floor)	- \forall , 2 , - \Rightarrow , 7
	Wet(Floor)	+ & , 7, 9 - \forall , 3, - \Rightarrow , 10
		- EI conclusion

سو عمومی را حذف کردیم و به جای x گذاشتم
b1 بعد از modus ponens استفاده کردیم
p proved
 $p \Rightarrow q$ proved
نتیجه میشه خط \wedge

Prove $(\forall x(P(x) \Rightarrow Q)) \Rightarrow (\exists xP(x) \Rightarrow Q)$

1.	$\forall x(P(x) \Rightarrow Q)$	
2.	$\exists xP(x)$	این قسمت را برای این از کل اثبات جدا کردیم زیرا از حذف سور وجودی که فرض کردیم به دست آمد
3.	$P(x)$	
4.	$P(x) \Rightarrow Q$	
5.	Q	
6.	Q	
7.	$\exists xP(x) \Rightarrow Q$	
8.	$(\forall x(P(x) \Rightarrow Q)) \Rightarrow (\exists xP(x) \Rightarrow Q)$	

Assumption

Assumption

- \exists Ass.

+ \forall , 1

- \Rightarrow 3, 4

- \exists Conc.

+ \Rightarrow 2, 6

+ \Rightarrow 1, 7

Prove $(\exists x P(x) \Rightarrow Q) \Rightarrow (\forall x (P(x) \Rightarrow Q))$

- | | | |
|----|---|----------------------|
| 1. | $(\exists x P(x) \Rightarrow Q)$ | Assumption |
| 2. | $\frac{}{P(x)}$ | Assumption |
| 3. | $\frac{}{\exists x P(x)}$ | + \exists |
| 4. | $\frac{}{Q}$ | - $\Rightarrow 1, 3$ |
| 5. | $P(x) \Rightarrow Q$ | - $\Rightarrow 2, 4$ |
| 6. | $\frac{}{\forall x (P(x) \Rightarrow Q)}$ | + $\forall 5$ |
| 7. | $(\exists x P(x) \Rightarrow Q) \Rightarrow (\forall x (P(x) \Rightarrow Q))$ | + $\Rightarrow 1, 6$ |

Unification

برای این که بررسی کنیم که ایا دو عبارت می توانند هم معنی باشند یا نه ؟
با ید بررسی کنیم که

آیا میشه این دو تا عبارت را جای متغیر هایشان می توان چیز هایی قرار داد که شکل دو عبارت یکی شود
که در resolution بسیار کاربرد دارد .

در منطق مرتبه اول ما متغیر داریم
یعنی x و y هایی داریم که object ها می توانند جای آن ها قرار بگیرند .

Can we find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ is such a substitution

$Unify(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Standardizing apart eliminates overlap of variables, e.g., $Knows(z_{17}, OJ)$

Unification

جای object ها نمی توان object دیگر قرار داد

Can we find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ is such a substitution

$\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Standardizing apart eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, OJ)$

Unification

Can we find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ is such a substitution

$\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Standardizing apart eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, OJ)$

Unification

Can we find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ is such a substitution

$\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}
Knows(John,x)	Knows(x,OJ)	

Standardizing apart eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, OJ)$

Unification

Can we find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ is such a substitution

$\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}
Knows(John,x)	Knows(x,OJ)	{fail}

Standardizing apart eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, OJ)$

Unification

To unify $\text{Knows}(\text{John}, x)$ and $\text{Knows}(y, z)$,

$$\theta = \{y/\text{John}, x/z\} \text{ or } \theta = \{y/\text{John}, x/\text{John}, z/\text{John}\}$$

The first unifier is **more general** than the second.

There is a single **most general unifier** (MGU) that is unique up to renaming of variables.

$$\text{MGU} = \{y/\text{John}, x/z\}$$

The unification algorithm

```
function UNIFY( $x, y, \theta$ ) returns a substitution to make  $x$  and  $y$  identical
  inputs:  $x$ , a variable, constant, list, or compound
           $y$ , a variable, constant, list, or compound
           $\theta$ , the substitution built up so far

  if  $\theta = \text{failure}$  then return failure
  else if  $x = y$  then return  $\theta$ 
  else if VARIABLE?( $x$ ) then return UNIFY-VAR( $x, y, \theta$ )
  else if VARIABLE?( $y$ ) then return UNIFY-VAR( $y, x, \theta$ )
  else if COMPOUND?( $x$ ) and COMPOUND?( $y$ ) then
    return UNIFY(ARGS[ $x$ ], ARGS[ $y$ ], UNIFY(OP[ $x$ ], OP[ $y$ ],  $\theta$ ))
  else if LIST?( $x$ ) and LIST?( $y$ ) then
    return UNIFY(REST[ $x$ ], REST[ $y$ ], UNIFY(FIRST[ $x$ ], FIRST[ $y$ ],  $\theta$ ))
  else return failure
```

The unification algorithm

```
function UNIFY-VAR(var, x,  $\theta$ ) returns a substitution
  inputs: var, a variable
          x, any expression
           $\theta$ , the substitution built up so far
  if  $\{var/val\} \in \theta$  then return UNIFY(val, x,  $\theta$ )
  else if  $\{x/val\} \in \theta$  then return UNIFY(var, val,  $\theta$ )
  else if OCCUR-CHECK?(var, x) then return failure
  else return add  $\{var/x\}$  to  $\theta$ 
```

Example Knowledge Base

Mark was a man. Mark was a Pompeian. All Pompeians were Romans. Caesar was a ruler. All Romans were either loyal to Caesar or hated him. Everyone is loyal to someone. People only try to assassinate rulers they are not loyal to. Mark tried to assassinate Caesar.

Example Knowledge Base

- (1) Mark was a man.
man(Mark)
یک relation به نام man اضافه می کنیم که مرد بودن یا انسان بودن را برای ما مشخص کند.
- (2) Was Mark loyal to Caesar?
Was Mark loyal to Caesar?
یک relation به نام loyal اضافه می کنیم که مرد بودن یا انسان بودن را برای ما مشخص کند.
- (3) Mark was a Pompeian.
Pompeian(Mark)
یک relation به نام pompeian اضافه می کنیم که مرد بودن یا انسان بودن را برای ما مشخص کند.
- (4) All Pompeians were Romans.
 $\forall x (\text{Pompeian}(x) \rightarrow \text{Roman}(x))$
- (5) Caesar was a ruler.
ruler(Caesar)
- (6) All Romans were either loyal to Caesar or hated him.
 $\forall x (\text{Roman}(x) \rightarrow \text{loyalto}(x, \text{Caesar}) \vee \text{hate}(x, \text{Caesar}))$
- (7) Everyone is loyal to someone.
 $\forall x \exists y \text{ loyalto}(x, y)$
- (8) People only try to assassinate rulers they are not loyal to.
 $\forall x \forall y (\text{person}(x) \wedge \text{ruler}(y) \wedge \text{tryassassinate}(x, y) \rightarrow \neg \text{loyalto}(x, y))$
- (9) Mark tried to assassinate Caesar.
tryassassinate(Mark, Caesar)

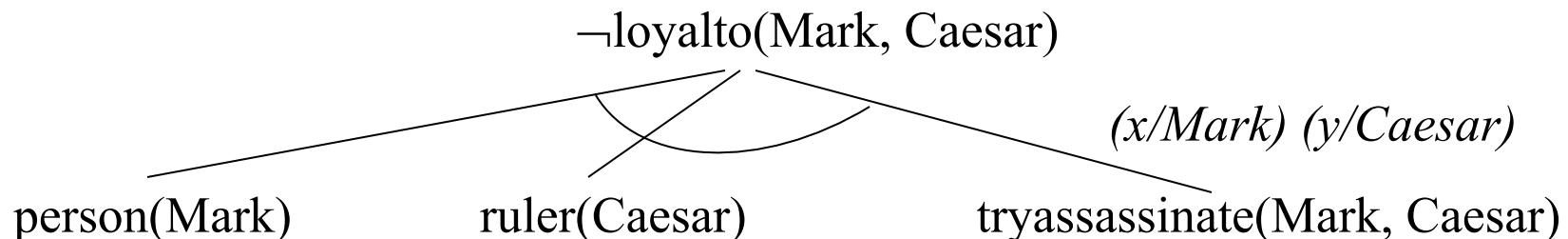
Reasoning Backward

- (1) $\text{man}(\text{Mark})$
- (2) $\text{Pompeian}(\text{Mark})$
- (3) $\forall x (\text{Pompeian}(x) \rightarrow \text{Roman}(x))$
- (4) $\text{ruler}(\text{Caesar})$
- (5) $\forall x (\text{Roman}(x) \rightarrow \text{loyalto}(x, \text{Caesar}) \vee \text{hate}(x, \text{Caesar}))$
- (6) $\forall x \exists y \text{ loyalto}(x, y)$
- (7) $\forall x \forall y (\text{person}(x) \wedge \text{ruler}(y) \wedge \text{tryassassinate}(x, y) \rightarrow \neg \text{loyalto}(x, y))$
- (8) $\text{tryassassinate}(\text{Mark}, \text{Caesar})$

حکم که می خواهیم اثبات کنیم با سمت راست عبارت ۷ قابل unify شدن می باشد
این کار را معمولا در reasoning backward انجام می دهیم و بعد
به دنبال ادامه مراحل می رویم.

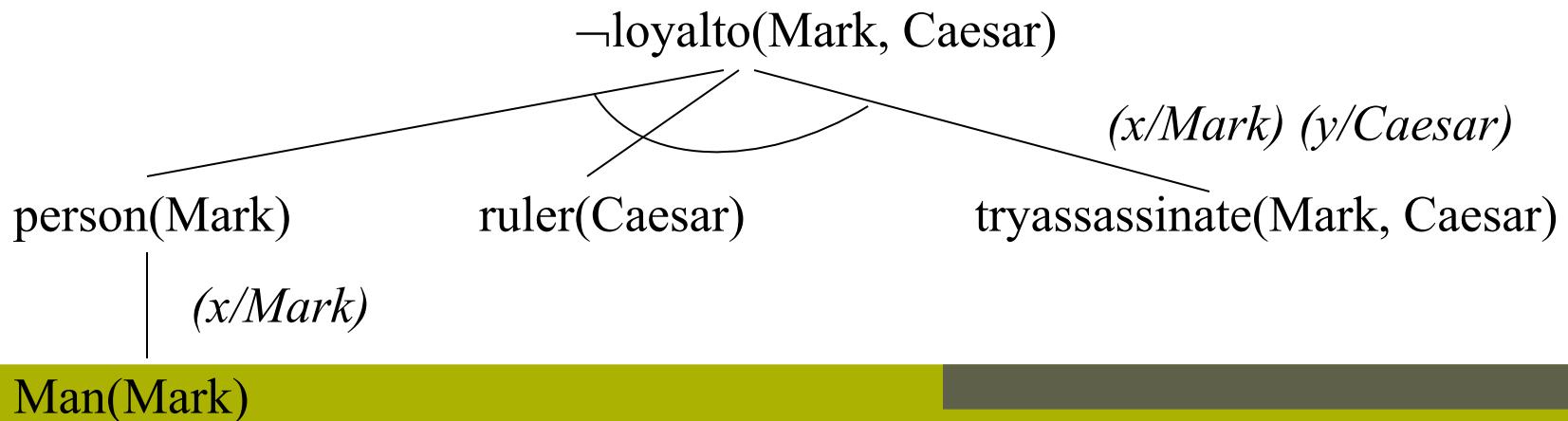
زمانی که unify می کنیم این تغییر در همه جا اعمال می شود.

الآن ما person(mark) را نداریم
بنابراین یک مشکلی وجود دارد
در واقع در KB ارتباطی بین man و person آورده نشده است 😞



Reasoning Backward

- (1) $\text{man}(\text{Mark})$
 - (2) $\text{Pompeian}(\text{Mark})$
 - (3) $\forall x (\text{Pompeian}(x) \rightarrow \text{Roman}(x))$
 - (4) $\text{ruler}(\text{Caesar})$
 - (5) $\forall x (\text{Roman}(x) \rightarrow \text{loyalto}(x, \text{Caesar}) \vee \text{hate}(x, \text{Caesar}))$
 - (6) $\forall x \exists y \text{ loyalto}(x, y)$
 - (7) $\forall x \forall y (\text{person}(x) \wedge \text{ruler}(y) \wedge \text{tryassassinate}(x, y) \rightarrow \neg \text{loyalto}(x, y))$
 - (8) $\text{tryassassinate}(\text{Mark}, \text{Caesar})$
 - (9) $\forall x (\text{man}(x) \rightarrow \text{person}(x))$
- اشکال اسلاید قبل را برطرف می کن. (قانون ۹)



Functions and Predicates

مثال ۲

(1) Mark was a man.

man(Mark)

چون اصول موضوعه حساب را می توان با منطق مرتبه اول به صورت کامل بازنمایی کرد می توانیم از عملگر (-) استفاده کنیم

(2) Mark was a Pompeian.

Pompeian(Mark)

(3) Mark was born in 40 A.D.

born(Mark, 40)

(4) All men are mortal.

$\forall x (\text{man}(x) \rightarrow \text{mortal}(x))$

(5) A volcano erupted in 79 A.D.

erupted(volcano, 79)

(6) All Pompeians died in 79 A.D.

$\forall x (\text{Pompeian}(x) \rightarrow \text{died}(x, 79))$

(7) No mortal lives longer than 150 years.

$\forall x \forall t1 \forall t2 (\text{mortal}(x) \wedge \text{born}(x, t1) \wedge \text{gt}(t2-t1, 150) \rightarrow \text{dead}(x, t2))$

(8) It is now 2010.

now = 2010

Is Mark alive?

Functions and Predicates

- (1) man(Mark)
- (2) Pompeian(Mark)
- (3) born(Mark, 40)
- (4) $\forall x (\text{man}(x) \rightarrow \text{mortal}(x))$
- (5) erupted(volcano, 79)
- (6) $\forall x (\text{Pompeian}(x) \rightarrow \text{died}(x, 79))$
- (7) $\forall x \forall t1 \forall t2 (\text{mortal}(x) \wedge \text{born}(x, t1) \wedge \text{gt}(t2-t1, 150) \rightarrow \text{dead}(x, t2))$
- (8) now = 2010

در اینجا دوباره مشکل داریم
زیرا ارتباط بین alive و dead را مشخص نکردیم

$\neg \text{alive}(\text{Mark}, \text{now})$

Functions and Predicates - Filling in the Blanks

- (1) man(Mark)
- (2) Pompeian(Mark)
- (3) born(Mark, 40)
- (4) $\forall x (\text{man}(x) \rightarrow \text{mortal}(x))$
- (5) erupted(volcano, 79)
- (6) $\forall x (\text{Pompeian}(x) \rightarrow \text{died}(x, 79))$
- (7) $\forall x \forall t1 \forall t2 (\text{mortal}(x) \wedge \text{born}(x, t1) \wedge \text{gt}(t2-t1, 150) \rightarrow \text{dead}(x, t2))$
- (8) now = 2010

سه قانون مشخص شده مشکل اسلاید قبل را حل میکند.

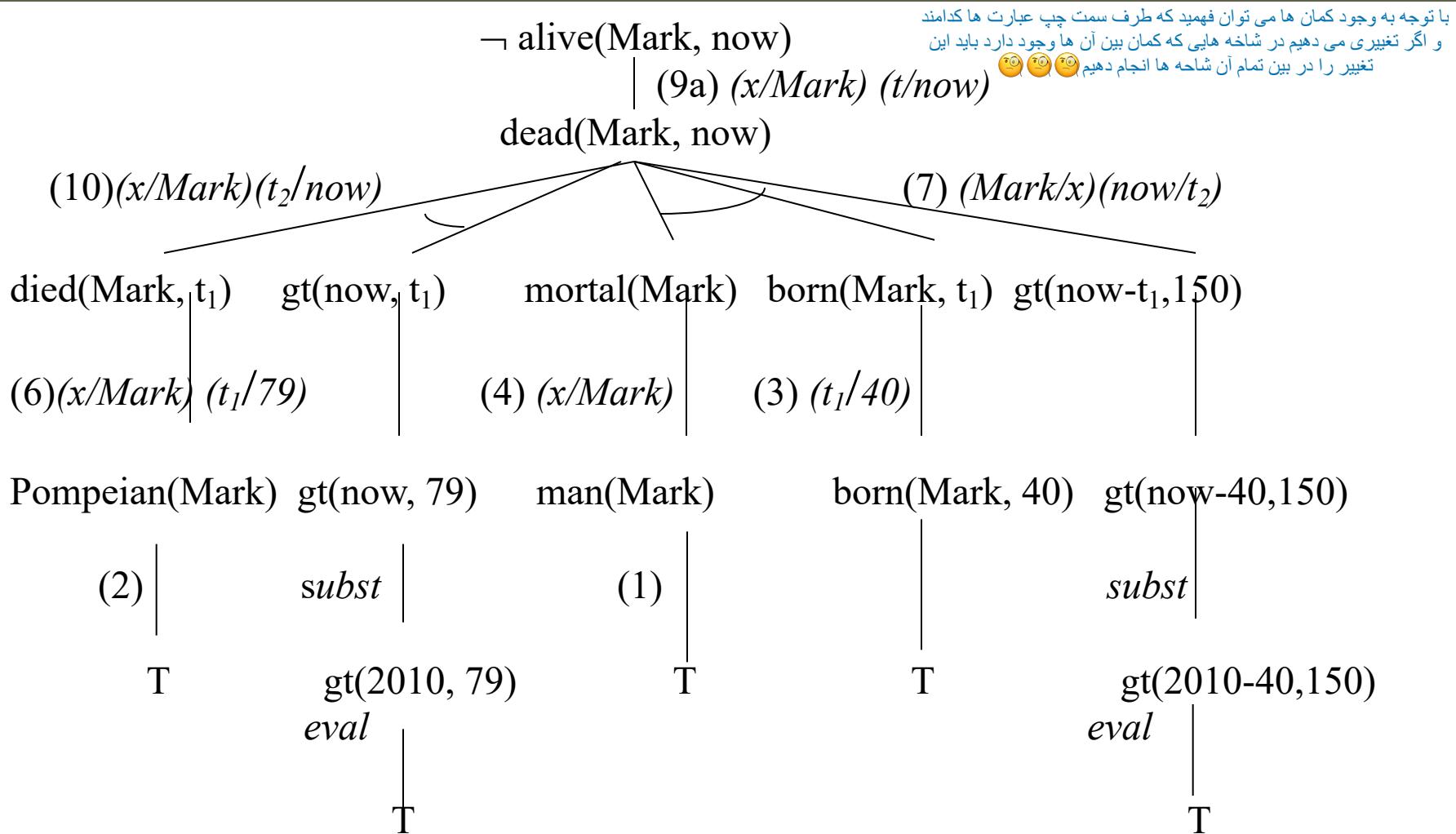
- (9a) $\forall x \forall t (\text{dead}(x, t) \rightarrow \neg \text{alive}(x, t))$
- (9b) $\forall x \forall t (\neg \text{alive}(x, t) \rightarrow \text{dead}(x, t))$
- (10) $\forall x \forall t1 \forall t2 (\text{died}(x, t1) \wedge \text{gt}(t2, t1) \rightarrow \text{dead}(x, t2))$

$\neg \text{alive}(\text{Mark}, \text{now})$

در اینجا یک مشکل وجود دارد : در اینجا ما ارتباط بین **dead** و **died** را نداریم

و از روی KB هم قابل برداشت نیست
که قانون ۱۰ این مشکل را حل می کند

Showing that Mark is Not Alive



A Harder One

Given:

$$\forall x [\text{Roman}(x) \wedge \text{know}(x, \text{Mark})] \rightarrow [\text{hate}(x, \text{Caesar}) \vee (\forall y (\exists z \text{ hate}(y, z)) \rightarrow \text{thinkcrazy}(x, y))]$$

$\text{Roman}(\text{Isaac})$

$\neg \text{hate}(\text{Isaac}, \text{Caesar})$

$\text{hate}(\text{Paulus}, \text{Mark})$

$\neg \text{thinkcrazy}(\text{Isaac}, \text{Paulus})$

Prove:

$\neg \text{know}(\text{Isaac}, \text{Mark})$

Clause Form Simplifies the Process

Standard:

$$\forall x [\text{Roman}(x) \wedge \text{know}(x, \text{Mark})] \rightarrow [\text{hate}(x, \text{Caesar}) \vee (\forall y (\exists z \text{ hate}(y, z)) \rightarrow \text{thinkcrazy}(x, y))]$$

Clausal:

$$\neg \text{Roman}(x) \vee \neg \text{know}(x, \text{Mark}) \vee \text{hate}(x, \text{Caesar}) \vee \neg \text{hate}(y, z) \vee \text{thinkcrazy}(x, z)$$

Conversion to Clause Form - Step 1

1. Eliminate \rightarrow , using the fact that $a \rightarrow b$ is equivalent to $\neg a \vee b$

$$\forall x [\text{Roman}(x) \wedge \text{know}(x, \text{Mark})] \rightarrow \\ [\text{hate}(x, \text{Caesar}) \vee (\forall y (\exists z \text{hate}(y, z)) \rightarrow \text{thinkcrazy}(x, y))]$$

$$\forall x \neg [\text{Roman}(x) \wedge \text{know}(x, \text{Mark})] \vee \\ [\text{hate}(x, \text{Caesar}) \vee (\forall y \neg(\exists z \text{hate}(y, z)) \vee \text{thinkcrazy}(x, y))]$$

Conversion to Clause Form - Step 2

2. Reduce the scope of each \neg to a single term, using:

- $\neg(\neg p) = p$
- deMorgan's laws
- $\neg \forall x P(x) \equiv \exists x \neg P(x)$
- $\neg \exists x P(x) \equiv \forall x \neg P(x)$

$$\forall x \neg [\text{Roman}(x) \wedge \text{know}(x, \text{Mark})] \vee \\ [\text{hate}(x, \text{Caesar}) \vee (\forall y \neg (\exists z \text{hate}(y, z)) \vee \text{thinkcrazy}(x, y))]$$

$$\forall x [\neg \text{Roman}(x) \vee \neg \text{know}(x, \text{Mark})] \vee \\ [\text{hate}(x, \text{Caesar}) \vee (\forall y \forall z \neg \text{hate}(y, z) \vee \text{thinkcrazy}(x, y))]$$

Conversion to Clause Form - Step 3

3. Standardize variables so that each quantifier binds a unique variable.

$$\forall x P(x) \vee \forall x Q(x)$$

$$\forall x P(x) \vee \forall y Q(y)$$

Conversion to Clause Form - Step 4

4. Move all quantifiers to the left without changing their relative order.

سورهارا به سمت چپ می آوریم با همان ترتیبی که در فرمول آمده است

$$\forall x [\neg \text{Roman}(x) \vee \neg \text{know}(x, \text{Mark})] \vee \\ [\text{hate}(x, \text{Caesar}) \vee (\forall y \forall z \neg \text{hate}(y, z) \vee \text{thinkcrazy}(x, y))]$$

$$\forall x \forall y \forall z [\neg \text{Roman}(x) \vee \neg \text{know}(x, \text{Mark})] \vee \\ [\text{hate}(x, \text{Caesar}) \vee (\neg \text{hate}(y, z) \vee \text{thinkcrazy}(x, y))]$$

At this point, we have prenex normal form.

Conversion to Clause Form - Step 5

5. Eliminate existential quantifiers through the use of Skolem functions and constants.

سورهای وجودی را حذف می کنیم

$\exists x \text{ Roman}(x)$

$\text{Roman}(\text{S1})$

$\forall x \exists z \text{ father-of}(x, z)$

$\forall x \text{ father-of}(x, \text{S2}(x))$

اگر حالت دومی باشد یک function گیریم که پارامتر های آن x می باشد
در صورتی که ۲۰ گزاره بود آن گاه کافی بود که تمام آن ۲۰ پارامتر را به تابع بدهیم.

Conversion to Clause Form - Step 6

6. Drop the prefix since all remaining quantifiers are universal.

سورهای عمومی را حذف می کنیم

$$\forall x \forall y \forall z [\neg \text{Roman}(x) \vee \neg \text{know}(x, \text{Mark})] \vee \\ [\text{hate}(x, \text{Caesar}) \vee (\neg \text{hate}(y, z) \vee \text{thinkcrazy}(x, y))]$$
$$[\neg \text{Roman}(x) \vee \neg \text{know}(x, \text{Mark})] \vee \\ [\text{hate}(x, \text{Caesar}) \vee (\neg \text{hate}(y, z) \vee \text{thinkcrazy}(x, y))]$$

Conversion to Clause Form - Step 7

7. Convert the matrix into a conjunction of disjuncts by using:

- Associative properties of \wedge and \vee .

$$[\neg \text{Roman}(x) \vee \neg \text{know}(x, \text{Mark})] \vee \\ [\text{hate}(x, \text{Caesar}) \vee (\neg \text{hate}(y, z) \vee \text{thinkcrazy}(x, y))]$$

$$\neg \text{Roman}(x) \vee \neg \text{know}(x, \text{Mark}) \vee \\ \text{hate}(x, \text{Caesar}) \vee (\neg \text{hate}(y, z) \vee \text{thinkcrazy}(x, y))$$

- Distribution of \vee over \wedge .

$$(P(x) \wedge Q(x)) \vee T(x)$$

$$(P(x) \vee T(x)) \wedge (Q(x) \vee T(x))$$

Conversion to Clause Form - Step 8

8. Create a separate clause for each conjunct.

$$(P(x) \vee T(x)) \wedge (Q(x) \vee T(x))$$

$$(P(x) \vee T(x))$$

$$(Q(x) \vee T(x))$$

Conversion to Clause Form - Step 9

9. Standardize apart the variables.

$$(P(x) \vee T(x))$$

$$(Q(x) \vee T(x))$$

کاری می کنیم متغیر های متفاوت شود .

$$(P(x) \vee T(x))$$

$$(Q(y) \vee T(y))$$

Resolution in Predicate Logic

$$\frac{p' \vee q \quad \neg p \vee r}{(q \vee r)\theta} \quad \text{where } p'\theta = p\theta$$

در اینجا می‌دانیم در هر صورت y یا r مقدار $p^{\prime\prime}$ true دارد. (بدون در نظر گرفتن $p^{\prime\prime}$)

$$\frac{p(x) \vee q(x) \quad \neg p(a) \vee r(b)}{(q(x) \vee r(b)) [x/a] \Rightarrow q(a) \vee r(b)} \quad \text{where } \theta = [x/a]$$

فرض می‌کنیم a و b مقادیر constatnt باشند

در اینجا ممکن است که نقیض p را داشته باشیم و p را نداشته باشیم ولی چیزی داشته باشیم که با p قابل unify شدن باشد
اینجا از استفاده می‌کنیم:
می‌گوییم unification را انجام بده و این دو را با هم حذف کن
ولی در جوابی که استنتاج می‌کنی باید آن unification را اعمال بکنی

All variables assumed universally quantified.

Importance of Standardizing Variables Apart

تأثیر آخرین استاندارد سازی: متغیرهای کلازهای مختلف را متفاوت را متفاوت می کردیم (مثال)

Consider:

1. $\neg\text{father}(x, y) \vee \neg\text{woman}(x)$
 $\{\text{father}(x, y) \rightarrow \neg\text{woman}(x)\}$
2. $\neg\text{mother}(x, y) \vee \text{woman}(x)$
 $\{\text{mother}(x, y) \rightarrow \text{woman}(x)\}$
3. $\text{mother}(\text{Chris}, \text{Mary})$
4. $\text{father}(\text{Chris}, \text{Bill})$

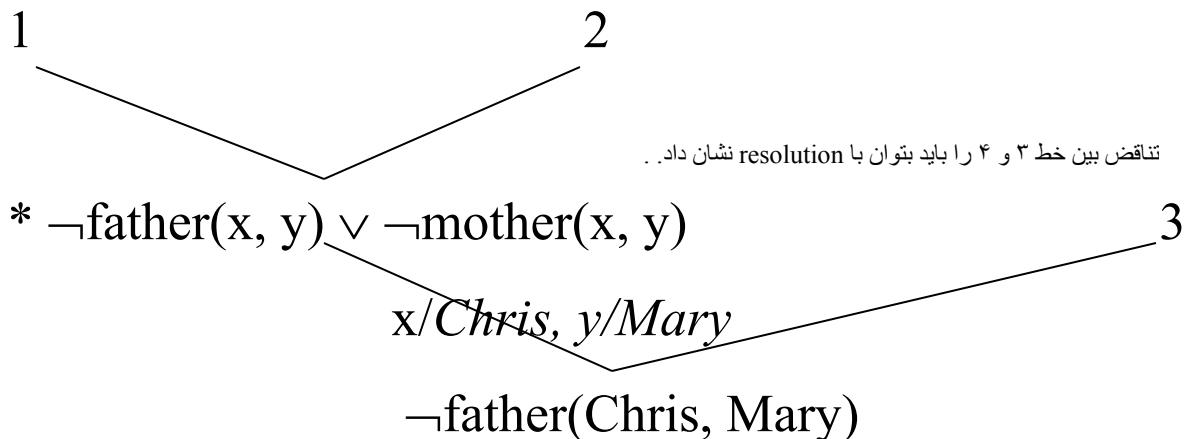
به ازای هر x و y (سورعومی برای x و y) وجود دارد و الان این جا نوشته نشده که مهم نیست زیرا در فرم نرمال نهایتاً هیچ سوری وجود ندارد

اتفاقی که می افتد این است که زمانی که این دو را با یک دیگر resolve می کنیم نتیجه اش نباید بشود عبارت * و اگر x در y باشد بنابراین x مادر هیچ کس نیست اشکال برای این به وجود آمد که y و x هر دو یکی بوند دو کلاز را به یکدیگر چسباندیم این دو y یکی هستند و ما یک تعهد اضافه ای اضافه کردیم

بنابراین حق نداریم در زمان resolution متغیرهار رو یک اسم بگیریم و این ها کلاس های متفاوتی هستند

مثلًا باید x و y بگیریم و با بعد resolution را انجام دهیم

این استنتاج یک اشکال دارد:
اشکال در معنا resolution اتفاق می افتد
و در خود resolution اشکالی وارد نمی شود

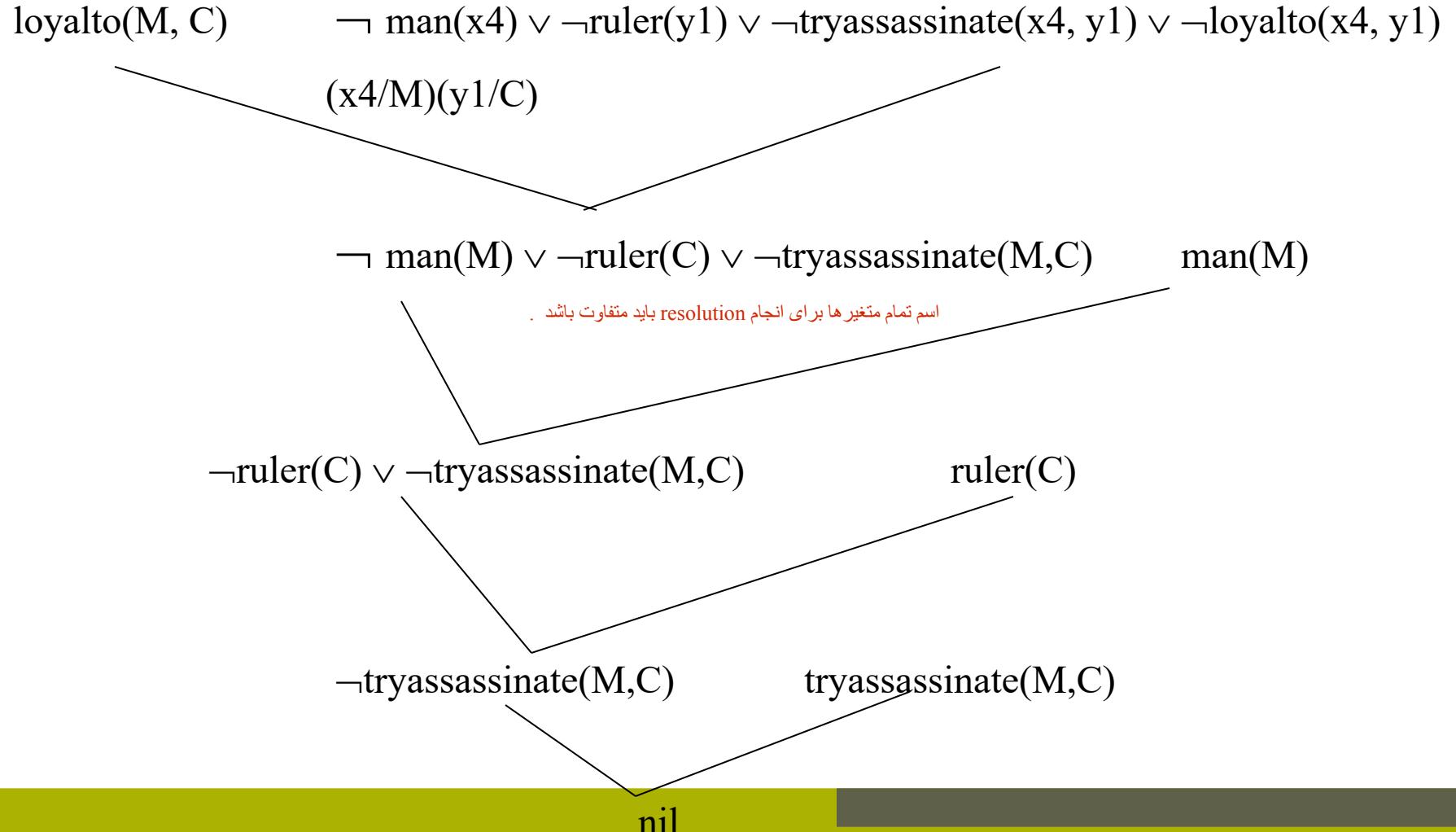


Back to Mark

- (1) $\text{man}(\text{Mark})$
- (2) $\text{Pompeian}(\text{Mark})$
- (3) $\neg\text{Pompeian}(x_1) \vee \text{Roman}(x_1)$
- (4) $\text{ruler}(\text{Caesar})$
- (5) $\neg\text{Roman}(x_2) \vee \text{loyalto}(x_2, \text{Caesar}) \vee \text{hate}(x_2, \text{Caesar})$
- (6) $\text{loyalto}(x_3, \text{S1}(x_3))$
- (7) $\neg\text{man}(x_4) \vee \neg\text{ruler}(y_1) \vee \neg\text{tryassassinate}(x_4, y_1) \vee \neg\text{loyalto}(x_4, y_1)$
- (8) $\text{tryassassinate}(\text{Mark}, \text{Caesar})$

همان مثال mike می باشد که با unification قرار است حل شود و در این اسلاید به صورت فرم نرمال کلزاں نوشته شده است.

Proving Mark Not Loyal to Caesar



Does Isaac Know Mark?

Given:

1. $\forall x [\text{Roman}(x) \wedge \text{know}(x, \text{Mark})] \rightarrow$
 $[\text{hate}(x, \text{Caesar}) \vee (\forall y (\exists z \text{ hate}(y, z)) \rightarrow \text{thinkcrazy}(x, y))]$

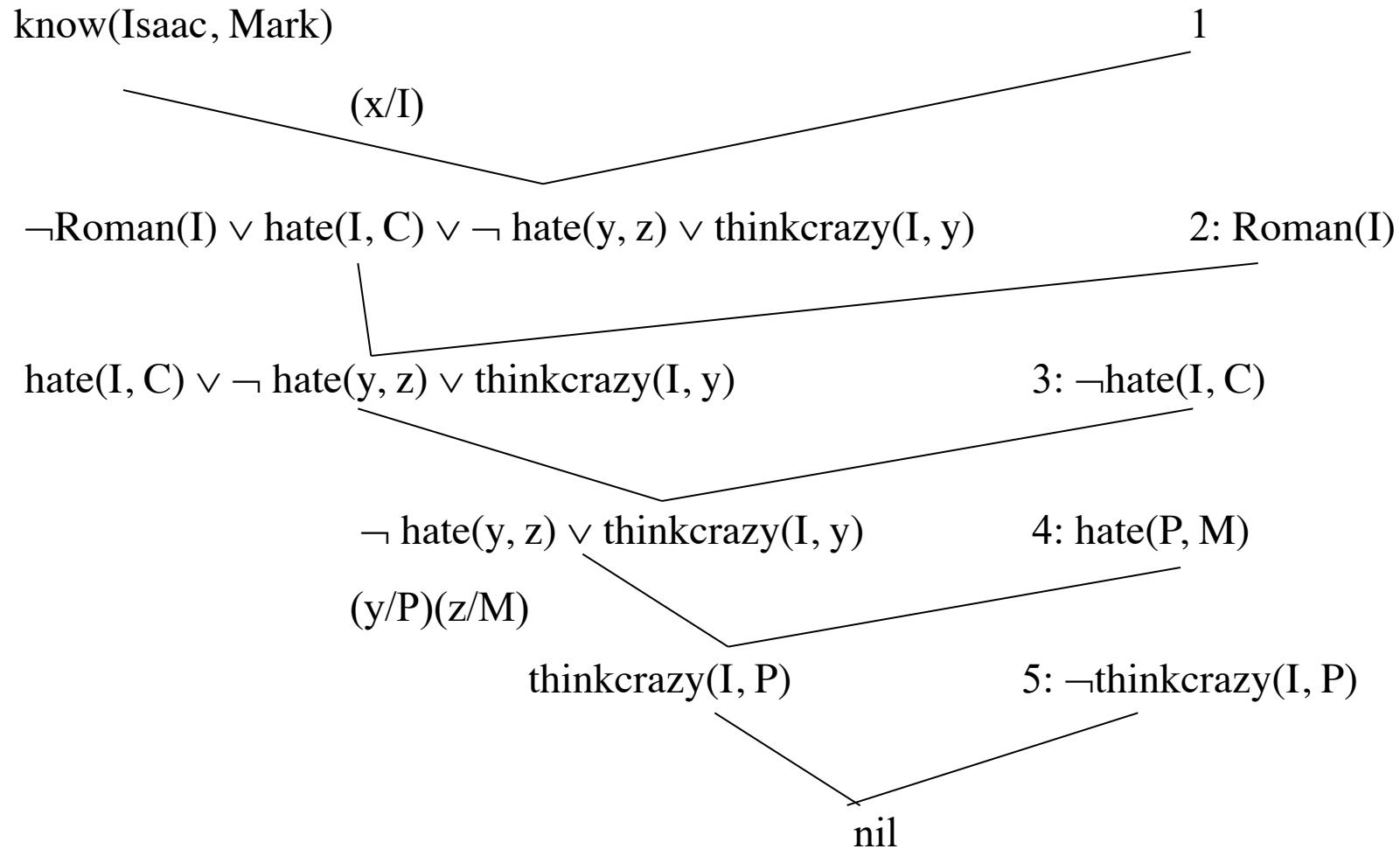
$$\neg \text{Roman}(x) \vee \neg \text{know}(x, \text{Mark}) \vee \text{hate}(x, \text{Caesar}) \vee \\ \neg \text{hate}(y, z) \vee \text{thinkcrazy}(x, y)$$

2. $\text{Roman}(\text{Isaac})$
3. $\neg \text{hate}(\text{Isaac}, \text{Caesar})$
4. $\text{hate}(\text{Paulus}, \text{Mark})$
5. $\neg \text{thinkcrazy}(\text{Isaac}, \text{Paulus})$

Prove:

$$\neg \text{know}(\text{Isaac}, \text{Mark}) \qquad * \text{ know}(\text{Isaac}, \text{Mark})$$

Proving Isaac Doesn't know Mark



Question Answering

When did Mark die?

$\text{died}(\text{Mark}, ??)$

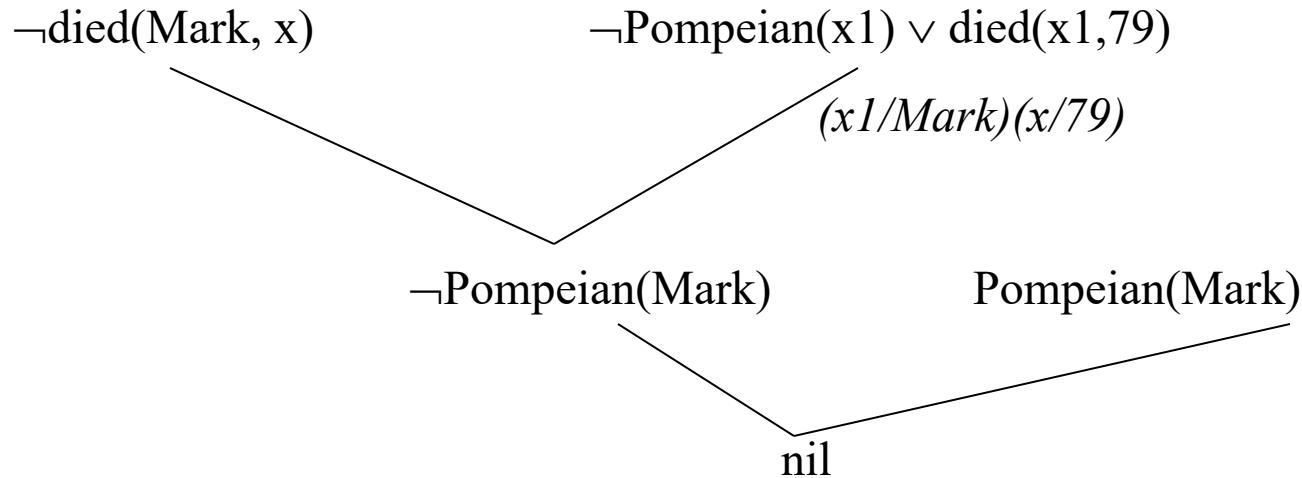
Mark must have died sometime, so we could try to prove:

$\exists x \text{ died}(\text{Mark}, x)$ by adding to the KB:

$\neg \exists x \text{ died}(\text{Mark}, x)$. Converting to clause form:

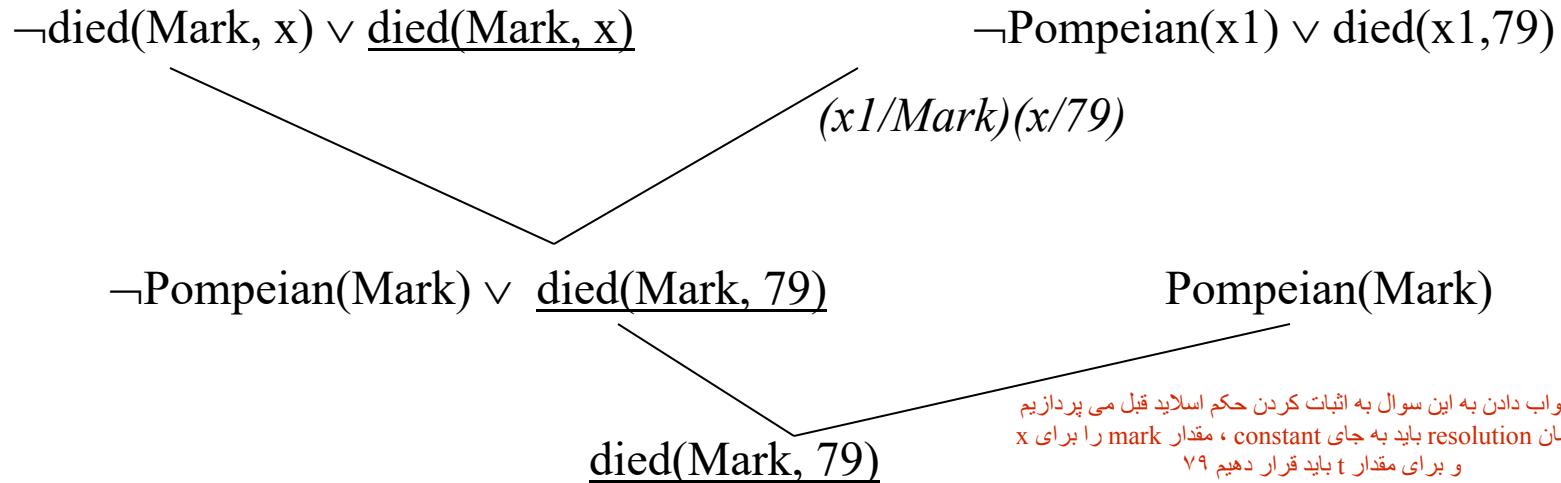
$\forall x \neg \text{died}(\text{Mark}, x)$

When Did Mark Die?



But how do we retrieve the answer 79?

When Did Mark Die?



برای جواب دادن به این سوال به اثبات کردن حکم اسلاید قبل می پردازیم
و در زمان resolution باید به جای constant ، مقدار mark را برای x
و برای مقدار t باید قرار دهیم ۷۹

The underlined literal will get carried along and collect all the substitutions. We stop when we generate a clause that contains only it.

Resolution Strategies

وقتی گزاره ای unit باشد یعنی فقط یک literal داخل آن وجود داشته باشد
اگر بتوانیم این گزاره با یک گزاره دیگر که شامل k لیترال دیگر است
نتیجه شامل k-1 کلاز خواهد داشت

Unit Preference:

هدف: رسیدن به کلاس تنه
و تنها راه ادغام کردن بک کلاس تکی با یک کلاس چند تایی

یعنی اگر کلاس یک دانه ای بدھیم اولویت داشته باشد برای انتخاب برای resolve شدن

- prefer to do resolutions where 1 sentence is a single literal, a **unit clause**
- goal is to produce the empty clause, focus search by producing resolvents that are shorter

*** complete but too slow for medium sized problems**

در این الگوریتم مافقط اولویت برقرار کردیم بین کلاس هایی که می خواهیم با یکدیگر resolve کنیم

Unit Resolution:

می خواهیم فقط زمانی resolution انجام دهیم که کلاز unit داشته باشیم

- requires at least 1 to be a unit clause

*** complete for Horn Clauses**

Unit Resolution Example (1)

1. $p \vee q$	KB
2. $\sim p \vee r$	KB
3. $\sim q \vee r$	KB
4. $\sim r$	S
5. $\sim p$	2, 4
6. $\sim q$	3, 4
7. q	1, 5
8. p	1, 6
9. r	3, 7
10. nil	6, 7
11. r	2, 8
12. nil	5, 8



این مثال دارای فرم نرمال horn نمی باشد !!!!!

Unit Resolution Example(2)

$$1. p \vee q$$

با استفاده از unit resolution نمی توان هیچ اثبات درستی نمی توان از این KB انجام داد !!!

$$2. \sim p \vee q$$

$$3. p \vee \sim q$$

$$4. \sim p \vee \sim q$$

نقیض تمام دو تایی ها آورده شده است .
و این ۴ حالت نمی توانند با هم درست باشند.

Cannot perform a single unit resolution since all clauses are of size 2!

Resolution Strategies (Cont.)

فرض می کنیم می خواهیم یک تناقض را در یک KB نشان دهیم
یک سری از کلاز های داخل KB را جدا کن هر بار می خواهیم resolution انجام دهیم
حتما باید یکی از کلاز هایی که resolve می کنی را از داخل SoS برداری و دیگری هر کلازی می تواند باشد
نتیجه به SoS اضافه می شود.

Set-of-Support (SoS):

- identify some subset of sentences, called SoS
- P and Q can be resolved if one is from SoS
- resolvent is added to the SoS
- common approach:
 - ~query is the initial SoS, resolvents are added
 - assumes KB is satisfiable

این SoS معمولا یک مجموعه کوچک است زیرا می خواهیم این استراتژی سرعت را بالاتر ببرد

این روش در حالت کلی کامل نیست

اگر شرط زیرا را رعایت کنیم کامل می شود:

KB - SoS

باید consistence باشد

یعنی ایجاد کار باید از این sos بیاید

یعنی مسئول ایجاد تناقض باید sos باشد

***complete if KB-SoS is satisfiable**

بهترین کاندید برای sos همان نقیض حکم است



دلیل این که نقیض حکم بهترین کاندید است: اگر قرار باشد KB خودش inconsistency باشد هر چیزی را می توان از آن اثبات کرد پس این فرض خوبی است که تناقض در خود KB نیست !!!



Set of Support resolution example

در این حالت همان طور که می توان دید در داخل sos هم q هست و هم نقیض $\neg q$ که خودش تناقض است
بنابراین اثبات تمام است

1.	$p \vee q$	KB	
2.	$\neg p \vee r$	KB	
3.	$\neg q \vee r$	KB	
4.	<u>$\neg r$</u>	S	<hr/>
5.	$\neg p$	2, 4	add to S
6.	<u>$\neg q$</u>	3, 4	add to S
7.	q	1, 5	add to S
8.	<u>p</u>	1, 6	add to S
9.	r	3, 7	
10.	nil	6, 7	
11.	r	2, 8	
12.	nil	5, 8	

Resolution Strategies (Cont.)

Input Resolution:

- P and Q can be resolved if at least one is from the set of original clauses, i.e., KB and \neg query
- ***complete for Horn Clauses**

Linear Resolution:

- a slight generalization of input resolution
- P and Q can be resolved if:
 - P is from the set of the original clauses, or
 - P is an ancestor of Q, in the proof tree

***complete**

روش Linear Resolution از روش input استفاده می کند ولی می توان نشان داد برای انواع KB ها کامل است

یک حالت خاص از استفاده از sos می باشد.
یک KB داریم و یک چیزی هم به عنوان نقیض حکم به آن اضافه کردیم
این مجموعه گزاره های ابتدایی (input) ما را نشان می دهد
هر وقت که می خواهیم resolution انجام دهیم
حتما یکی از این گزاره های باید از داخل این مجموعه بیاید

نمیشه هردو کلازی که میخواهیم
کنیم جدید باشد .

کامل نیست
اگر مثل کامل نبودن
unit resolution
در نظر بگیریم
می بینیم در هر بار
resolution حداقل
یکی از گزاره هایی
داریم برای
resolution استفاده
می کنیم
دو تایی هستند
پس هیچ وقت نمی توان
به گزاره تنهی رسید
زیرا مثلا هر بار یک
گزاره یک دانه ای را با
یک گزاره دو تایی
می کنیم یک
گزاره تکی به دست می
آوریم
Input Resolution

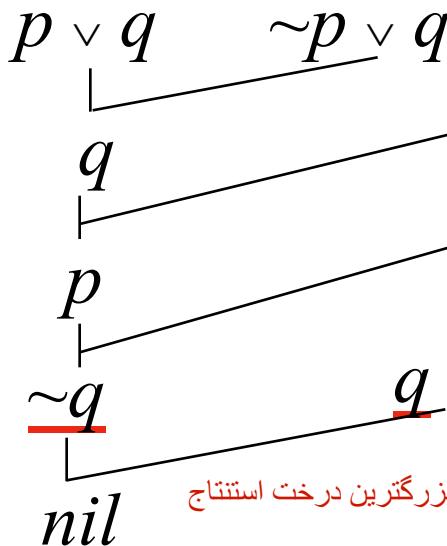
Input Resolution

Ex:

- | | | |
|----|----------------------|------|
| 1. | $p \vee q$ | KB |
| 2. | $\sim p \vee q$ | KB |
| 3. | $p \vee \sim q$ | KB |
| 4. | $\sim p \vee \sim q$ | KB |

**Cannot perform
input resolution!**

Linear Resolution



هر resolution که انجام می دهیم و یک کلاز به دست می آوریم
اجداد آن را هم نگه داریم
یعنی بدانیم کلاز نهایی از resolve کردن کدام کلاز ها به دست آمده
همچین درختی تولید می شود.

این resolution مجاز است اگر یک طرفش از input امده باشد
یا یک طرفش والد طرف دیگر باشد

نسبت به input خطی است اگر فرض کنیم اندازه (عمق) بزرگترین درخت استنتاج

constant

باشد (مطمئن باشیم که اثبات های خیلی بزرگی نداریم این روش بسیار به ما کمک می کند).

چون نقیض q از خود q به دست آمده ما می توانیم این دو را resolve کنیم.

حدودیتی که اضافه کردیم : یعنی یک گزاره فقط می تواند با پدرانش resolve بشود یا با ورودی

Natural Deduction vs. Resolution

- CNF is simpler to work with
- But, we lose readability and some useful control information

$$\forall x (\text{judge}(x) \And \neg \text{crooked}(x) \rightarrow \text{educated}(x))$$
$$\neg \text{judge}(x) \Or \text{crooked}(x) \Or \text{educated}(x)$$