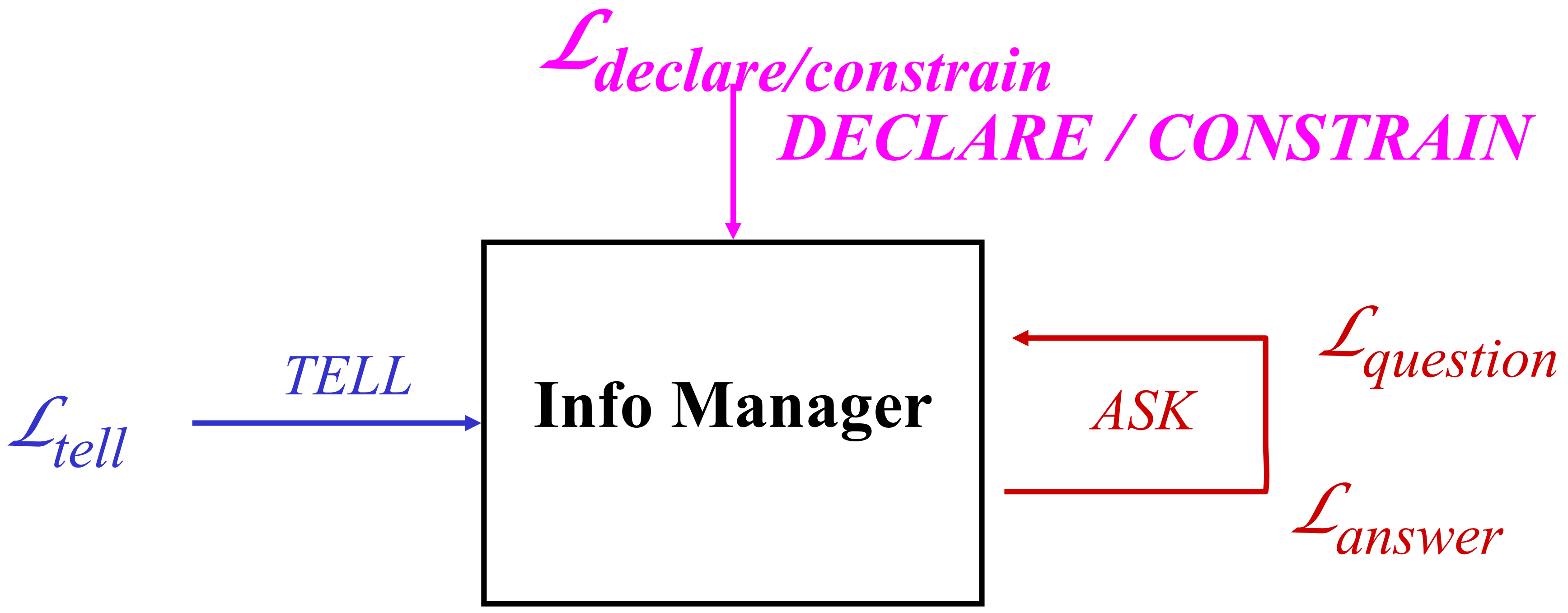
DATALOG: Logic-based Information Management

Extended view of Information Manager



- ***DECLARE*** : $\mathcal{L}_{declare} \times \mathbf{IM} \rightarrow \mathbf{IM}$
- *TELL*: $\mathcal{L}_{tell} \times \mathrm{IM} \rightarrow \mathrm{IM} \cup$ ***Exceptions***
- *ASK*: $\mathcal{L}_{question} \times \mathrm{IM} \rightarrow \mathcal{L}_{answer}$

Datalog$_0$

Example: **want to manage information about family relationships**

- ***Tell* <u>facts</u> like** "parents of chuck are liz and phil; chuck is male;"
- ***Ask* <u>questions</u> like** "Is chuck male? Is phil a father? Who is chuck's mother? Who are all the males?"

- ***predicates/relations*:** `male, parents`
- ***constants*:** `ed, vicky,...` ***(must start with lowercase)***
- ***ground atomic formula*:** *<pred>* **(** *<constt>* **,** … **) .**
- $\mathcal{L}_{tell}$ **= ground atomic formulas**

```
male(phil).      female(liz).
male(chuck).     female(di).

parents(chuck,liz,phil).
parents(ann,liz,phil).
parents(andy,di,chuck).
%% parents(child,mother,father)
```

```
male(phil).       female(liz).
male(chuck).      female(di).

%%  parents(child,mother,father)
parents(chuck,liz,phil).
parents(ann,liz,phil).
parents(andy,di,chuck).
```

$\mathcal{L}_{question} = \mathcal{L}_{tell}$

$\mathcal{L}_{answer} = \{\text{yes},\text{no}\}$

We can (sort of) use the Prolog programming language as a Datalog IM implementation.

```
?- male(phil).                  yes
?- female(chuck).               no
?- male(andy).                  no
?- parents(chuck,liz,phil).     yes
?- parents(liz,chuck,phil).
```

?

(In lectures I will be using an on-line PROLOG interpreter: http://swish.swi-prolog.org)

Datalog$_{0.25}$: queries with variables

- ***variables:*** `Y, A, Who` *(by convention, start with caps)*
- ***argument:*** ***constant*** **or** ***variable***
- ***atomic formula:*** *<predicate>* **(***<argument>* **,…) .**

$\mathcal{L}_{question}$ = **atomic formulas** (may have variable)

$\mathcal{L}_{qnswer}$ = {**yes,no**} or **variable substitutions**

```
likes(eve,pie).   person(tom).
likes(al,eve).    food(pie).
likes(eve,tom).   food(apple).
likes(eve,eve).
```

```
?-likes(al,Who).
  Who=eve
?-likes(eve,W).
  W=pie   ;
  W=tom   ;
  W=eve
?-likes(A,B).
  A=eve,B=pie ;
   A=al,B=eve ;
   ...
?-likes(A,A)
  D=eve
```

Transition from Datalog to SQL

Datalog:

```
likes(eve,pie).   person(tom).
likes(al,eve).    food(pie).
likes(eve,tom).   food(apple).
likes(eve,eve).
```

```
?-likes(al,Whom).
```

Corresponding SQL:

```
CREATE TABLE likes(who VARCHAR; whom VARCHAR;
                    PRIMARY KEY(who,whom));
INSERT INTO likes VALUES (‘eve’,’pie’);
...
```

```
SELECT t.whom
FROM likes t
WHERE t.who=‘al’;
```

Datalog$_{0.5}$: conjunctive queries

```
likes(eve,pie).   person(tom).
likes(al,eve).    food(pie).
likes(eve,tom).   food(apple).
likes(eve,eve).
```

$\mathcal{L}_{question}$ = **conjunction of atomic formulas**

'and' is written as comma ',' in Prolog

```
?-likes(eve,W) , person(W).
  W=tom
?-likes(eve,V) , likes(al,V).
  V=eve
?-likes(eve,W),likes(al,V),V=W.
  V=eve
?-likes(eve,W),person(W),food(V)
  W=tom,V=pie   ;
   W=tom,V=apple
?-likes(eve,W),likes(W,V).
  W=eve,V=pie ;
   W=eve,V=tom ;
   W=eve,V=eve
```

Datalog *RULES*

a) Rules as shorthand for some queries

```
likes(eve,pie).   person(tom).
likes(al,eve).    food(pie).
likes(eve,tom).   food(apple)
likes(eve,eve).
```

```
q1 :- likes(eve,V), person(V).
```

```
?-q1.
  yes
```

"Is there someone whom Eve likes?" ***(hides uninteresting vars)***

```
q2(Who):- likes(Who,F), food(F).
```

```
?-q2(H).
  H=eve
```

"Who likes some food?

(hides some variables not of interest -- the food F liked by that person)

DatalogNeg = Datalog$_{0.7}$ + Negation

```
likes(eve,pie).   person(tom).
likes(al,eve).    food(pie).
likes(eve,tom).   food(apple).
likes(eve,eve).
```

```
?- likes(eve,W), NOT likes(al,W).

  W=pie ; W=tom

?- NOT likes(_,Y).

        UNSAFE QUERY!
```

```
thing(X):- likes(X,_).

thing(X):- likes(_,X).

thing(X):- food(X).

thing(X):- person(X).

disliked(Y) :- thing(Y), NOT likes(_,Y).
```