Carnegie Mellon University

Pittsburgh, PA – 6 July 2006

The Deductive Spreadsheet

Iliano Cervesato

Deductive Solutions

iliano@deductivesolutions.com

The Traditional Spreadsheet

- Benefits
  - Simple access to complex *numerical* calculations
    - Intuitive interface
    - No formal training needed
    - Gentle learning curve
  - Effective decision support for *numerical* data
    - Financial analysis, budgets, grades, inventories, ...

Arithmetic

  - Ubiquitous
    - Over 50M users
    - Only recently surpassed by web browsers and mailers
- Opportunities
  - Simple access to **symbolic** calculations/reasoning
  - Effective decision support for **symbolic** data

Logic

Objectives of this Work

- Extend the spreadsheet with symbolic reasoning
  - Support symbolic decision-making
  - Provide functionalities to manipulate data symbolically
    - Logical language
    - Operational interpretation
    - Interface commands
  - Same ease of use as traditional spreadsheet
- Seamless integration into current model
  - Not a separate application

Results

- Extension of the traditional spreadsheet with:
  - Expressions over first-class tabular data
    - Datalog with negation, constraints, calculated values, lists
    - Equational relational algebra (extended)
  - Like database, but queries results permanently displayed
  - Efficient evaluation and update propagation
    - Guaranteed termination
  - Explanation facilities
- Extended user interface
- Good feedback from preliminary user testing

Rest of this Talk

- Requirements
- What is a spreadsheet?
- Extended core functionalities
  - Relational/Logical expressions
  - Evaluation / Updates / Explanation
- Extended user interface
  - Design methodology
  - Extensions
- User testing

Historical Attempts

- 1982: LogiCalc [Kriwaczek]
  - Spreadsheet in MicroProlog

  + relational views, integrity constraints, bidirectional variables, symbolic manipulations, complex objects
  - Teletype interface
- 1986: [van Emden]
  - Incremental queries, exploratory programming
- 1989: PERPLEX [Spenke & Beilken]
  - Bidirectional integrity constraints
- Then not much … until now!

Requirements

- Functional
  - Extension to core functionalities
- Cognitive
  - How they are made available to the user

Functional Requirements

- Conservativity
  - Retain all current functionalities
  - Current users incur no penalty
- Expressiveness
  - Datalog or better
- Supported inferences
  - Logical consequences
  - Explanation
- Termination
  - Always terminate
  - Timely
- Updates
  - Immediately propagated
- Integration
  - Deductive expressions within traditional formulas
  - Traditional formulas within deductive expressions

Cognitive Requirements

- Conservativity
  - Current commands do not change
  - Current users incur no penalty
- Consistency
  - Extended commands resemble traditional commands
  - Easy to learn
- Integration
  - Intuitive support for using deductive and traditional feature together
- Discovery
  - User gets skilled through usage
  - User learns by poking around

- Target audience: advanced and intermediate users

What is a Spreadsheet?

Mathematical model for

- Scalar spreadsheet
- Array formulas
- Relational support

Scalar Spreadsheets

A simple functional language without recursion

- 16,777,216 glorified calculators

- Functionalities
  - Input
    - Cells, Expressions
  - Calculate
    - Turn entered expressions into displayed values
  - Update
    - Propagate changes
  - Explanation (audit)
    - Catch errors

Spreadsheet Model

- Scalar expressions
  - A2 * 9/5 + 32
- Spreadsheet: s : Cell → Expr
  - No circular references
- Dependency graph: DG_s
  - Representation of s that highlights cell dependencies

Evaluation

Environment: Env = Cell → Val

Evaluation: eval: s → Env

- Best performed on dependency graph
  - Fixpoint calculation
    - Starts from undefined environment
    - # iterations = longest path in DG_s
  - Cost = **O**(used_cells)
    - Under semi-naïve strategy

Updates

- Determine tainted cells
  - Using dep. graph
- Evaluation starting from tainted environment
- Cost = **O**(tainted_cells)
  - Under semi-naïve strategy

Explanation

Why does A2 show 212?

- Commands to navigate DG_s from given cell
  - Highlight cells on which A2 depends
  - ... and those on which they depend
  - ... and those on which they depend
  - ... and those on which they depend
  - ...

Array Formulas

- Expressions associated to a block of cells
  - A44 := SUM(A2:A43)/42
  - B2:B43 := A2:A43 * 9/5 + 32

  s : Partition(Cell) → ArrayExp
- Map to scalar formulas
  - No circularity at that level
  - Inherit evaluation and update
- Immature user interface

Relational Support

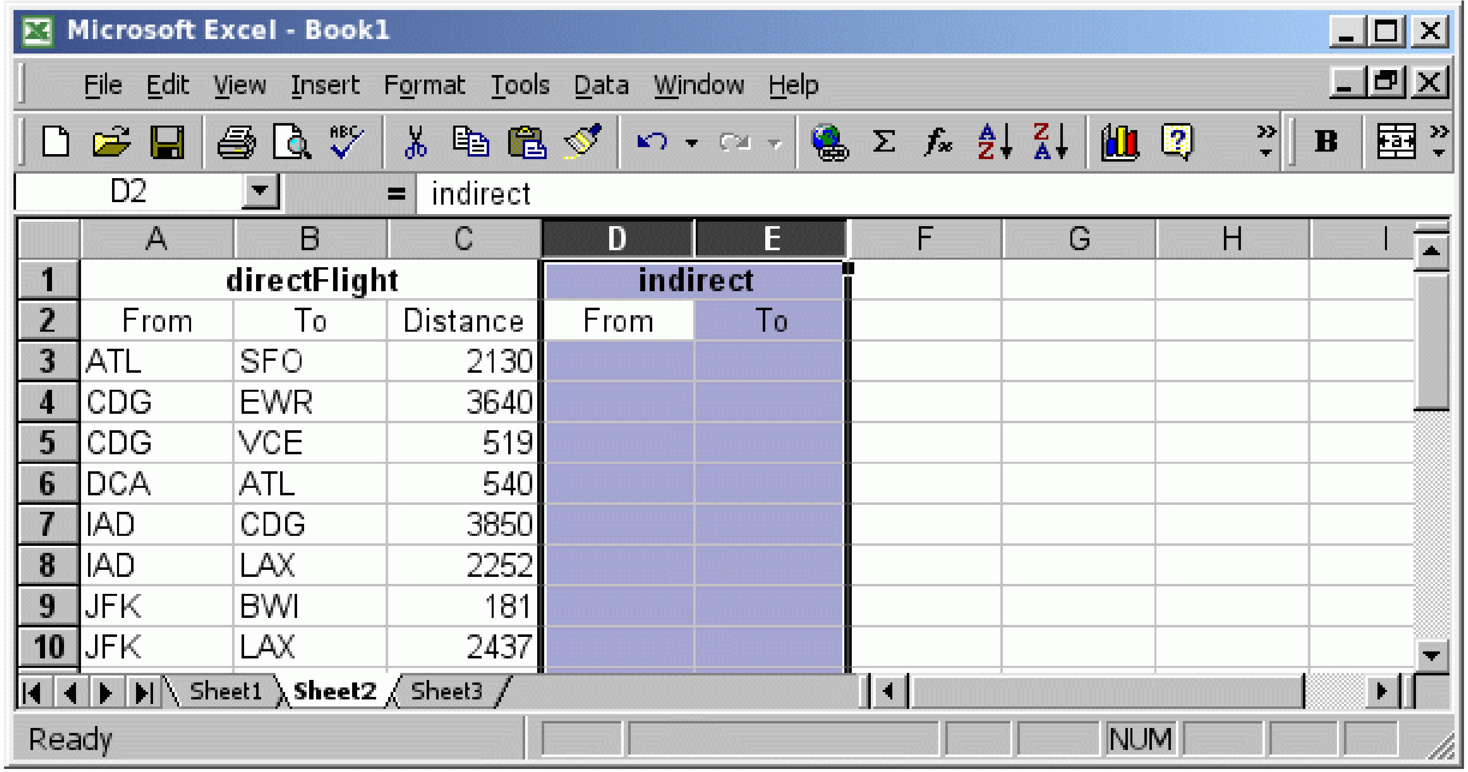
- “Data List” / “Databases” / ...
  - Minimal support for manipulating tabular data
    - Insertion wizard
    - Sorting
    - Selection
    - Import from other applications
  - Second class-objects
    - Functionalities as commands, not operations
  - No functions over multiple tables
    - No join

The Deductive Engine

- First-class relations
  - Relational expressions
  - Integration
- Logical counterpart
  - Datalog without recursion
  - Logical updates
  - Explanation as proof-search
- Deductive spreadsheet
  - Recursion
  - Bounded termination

Relations

- Interpret rows as records, columns as attributes
  - Or the other way around



D2 = indirect

	A	B	C	D	E	F	G	H	I
1	directFlight			indirect					
2	From	To	Distance	From	To				
3	ATL	SFO	2130						
4	CDG	EWR	3640						
5	CDG	VCE	519						
6	DCA	ATL	540						
7	IAD	CDG	3850						
8	IAD	LAX	2252						
9	JFK	BWI	181						
10	JFK	LAX	2437						

- Nothing new

Relational Expressions

- Associated to cell blocks
  - Like array formulas
- Manipulate relations as a whole
  - Union, difference, projection, selection, join
    - *Show all flights between Delta hubs less than 500 miles apart*

      $\pi_{\text{hub1.City,hub2.City}}\ \sigma_{\text{directFlight.Distance<500,hub1.Airline=“Delta”, hub2.Airline=“Delta”}}\ \text{hub1} \infty_{\text{City=From}}\ \text{directFlight} \infty_{\text{To=City}}\ \text{hub2}$
    - directFlight and hub could be calculated
  - Minor extension for calculated projection attributes
  - Result is treated as a set
    - Non-deterministic ordering
    - No duplicates

Interface to Usual Formulas

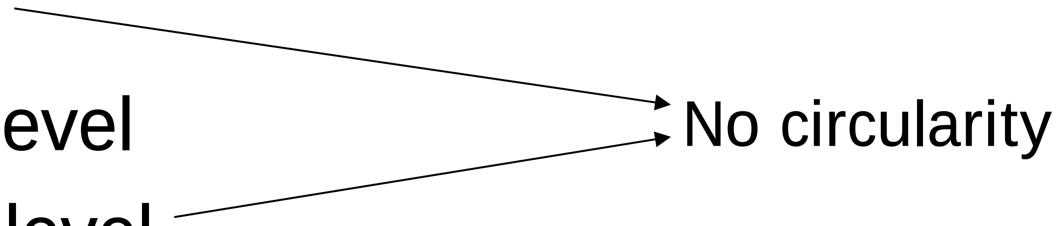
- Coercion from (array) formula to relation
  - <e> : compute e and interpret it as a relation
- Coercion from relational exp. to (array) formula
  - [r] : compute r and interpret it as an array
    - Ordering is non-deterministic
  - Add SORT as a new array operation
- Traditional formulas also in selection/projection attributes

- Relational expressions can appear within formulas
- Formulas can appear within relational expressions

Relational Spreadsheet

s : Partition(Cell) → ArrayExp U RelExp

- Cannot be reduced to scalar spreadsheet
- Several notions of dependency graph
  - Cell level
  - Relation level
  - Attribute level



No circularity

Functionalities

- Evaluation
  - Env = Partition(Cell) → Val U RelVal
  - Eval : s → Env
  - Cost = $\mathbf{O}(\text{records}^{\text{max\_join}})$
    - Semi-naïve evaluation
- Update
  - Identifies added/removed records
  - Start reevaluation from those
- Explanation
  - Similar to traditional spreadsheet
  - Inadequate

Logical Interpretation

- Rel. algebra equivalent to recursion-free Datalog

  Show all flights between Delta hubs less than 500 miles apart

  ```
  shortDeltaFlight(From,To) ←
          directFlight(From,To,Dist) & Dist < 500 &
          hub(From, “Delta”) & hub(To, “Delta”)
  ```

- Body literals can be negated
  - Stratified Datalog
- Extension with constraints
  - Generic
  - Head: operate on head-only variable
- Variables subject to safety restrictions

So What?

Harness wide array of logical tools

- 40 years of logic programming

- Logical interpretation of
  - Evaluation
    - Logical inference
  - Updates
    - Optimized evaluation
  - Explanation
    - proof-search

Evaluation Revisited

- Logical consequences computed as
  - Fixpoint of functional on logical interpretations
  - Bottom-up evaluation of logic programs
- Terminating
  - Fast strategies
    - Semi-naïve strategy
  - Used in deductive databases
- Scales to
  - Stratified negation
  - Safe constraints
  - Surrounding scalar/array formulas

Updates Revisited

- Incremental evaluation at heart of semi-naïve strategy
  - Optimization
- Adapts smoothly to generic updates
  - Positive updates
  - Negative updates

Explanation Revisited

- Display argument for computed record
  - Proof search
  - Top-down evaluation of logic programs
- Flexible explanation mechanism
  - *Why is this record there?*
  - *Why isn’t this record there?*
  - May contain variables
  - Proof of generic queries

The Deductive Spreadsheet

- Allow recursion
  - Subject to stratification

Show all pairs of cities connected by air

indirect(From,To) ← directFlight(From,To,_).

indirect(From,To) ← directFlight(From,Mid,_) & indirect(Mid,To)

- Strictly more expressive
  - Opens the door to a whole new class of problems
  - Even more so by exploiting spreadsheet environment
    - Overlapping traditional formulas

Examples of Expressiveness

- Any relational expression
  - Any SQL query
- Recursive queries
  - Transitive closure problems
    - Path in a graph
    - Travel planning
    - Hierarchies
    - Course requirements
    - Readiness of troops, ...
  - Bill of Material problem
  - Workflow problem
  - Meeting planner
  - Anti-trust problem

Extensions

- Head constraints in recursive clauses

  Show distance of trip

  indirect(From,To,Dist) ← directFlight(From,To,Dist).

  indirect(From,To,Dist) ← directFlight(From,Mid,Dist') & indirect(Mid,To,Dist'') & Dist = Dist'+ Dist''

  - Non-terminating in general
  - Put user-defined bound on recursion for these clauses
- Flat lists

  Show itinerary

  indirect(From,To,[From,To]) ← directFlight(From,To,_).

  indirect(From,To,[From,Mid|Rest]) ← directFlight(From,Mid) & indirect(Mid,To,[Mid|Rest])

  - Treated in the same way
- Embedded implication

The User Interface

- Design methodology

- Initial design
  - Most modern spreadsheets have nearly identical interfaces
  - Generic deductive extension
    - Demonstrated on Excel 2000

Interface Design Methodology

- Traditional approaches
  - Experts design user interface
    - We are not HCI experts
  - Refined through extensive user testing
    - No time/resources at this stage
- Lightweight approximate methods
  - Meant for application designers
  - Provide vocabulary for concepts and objectives
  - Obtain adequate first-cut
    - Validate/refine later using traditional approaches

Cognitive Dimensions

- "Discussion tools" for cognitive concepts
  - Viscosity
  - Consistency
  - Hard mental operations
  - Hidden dependencies, ...
- Vocabulary to make decisions
  - Evaluate cognitive effect
  - Plan trade-offs
- Scales to make rough measurements

Attention Investment Model

- Psycho-economic model to anticipate user behavior
  - Embracing novelty = investment of attentional effort
  - Will do if perceived pay-off > perceived risk
- Pay-off: larger class of solvable problems
- Costs:
  - Shifting to logical/relational mindset
  - Learning new syntax
- Risk: problem still not solvable
- Target audience
  - Needed skills
    - Tabular information, select cell ranges, comfortable with formulas
  - Advanced and intermediate users

Deductive Layout

- Nearly unchanged
  - No cognitive penalty
- Couple of new context-sensitive menu items
  - “Define Relation ...”
    - Give names to relation and attributes
    - Insert it in “defined predicates” list
    - Insert captions
  - “Explain”
  - Graphical construction of formulas

Textual Language of Formulas

Two alternatives

- Gives flexibility to user

- Embellished Datalog

```
indirect(From,To) IF directFlight(From,To,_).
indirect(From,To) IF directFlight(From,Mid,_) AND indirect(Mid,To)
```

- SQL-like language

```
indirect(To,From) = directFlight UNION
   SELECT directFlight.From, indirect.To FROM directFlight, indirect
   WHERE directFlight.To = indirect.From
```

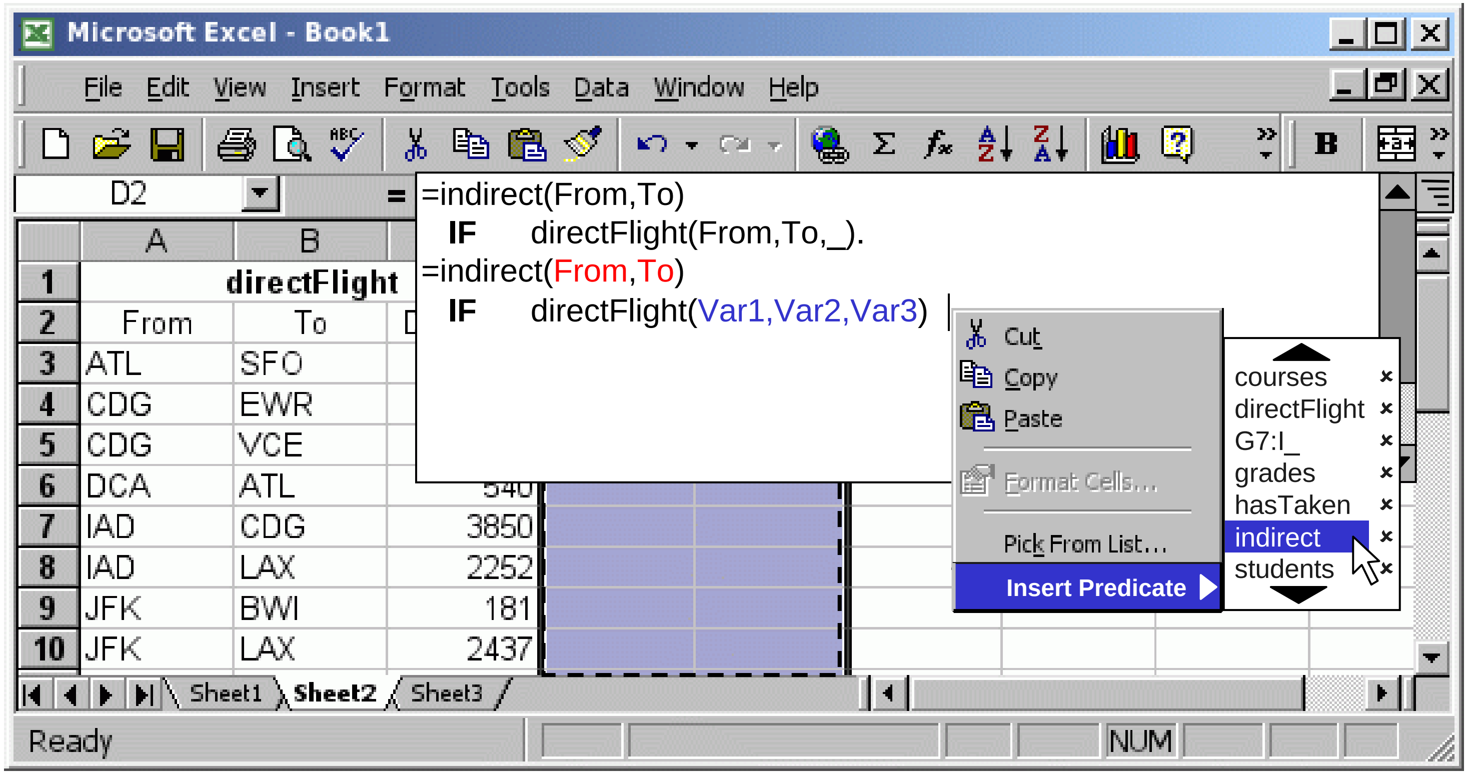
- Final choice to be guided by user feedback

Entering Formulas

- Typing in the formula bar
  - Syntax check “as-you-type”
    - Visual feedback
  - Autoformat
  - Precise error reporting
- Clicking around
- Wizards
- Cut and paste

Mouse-Assisted Definition

- Construct formula with a few mouse clicks
  - Names from “predicate list” or spreadsheet



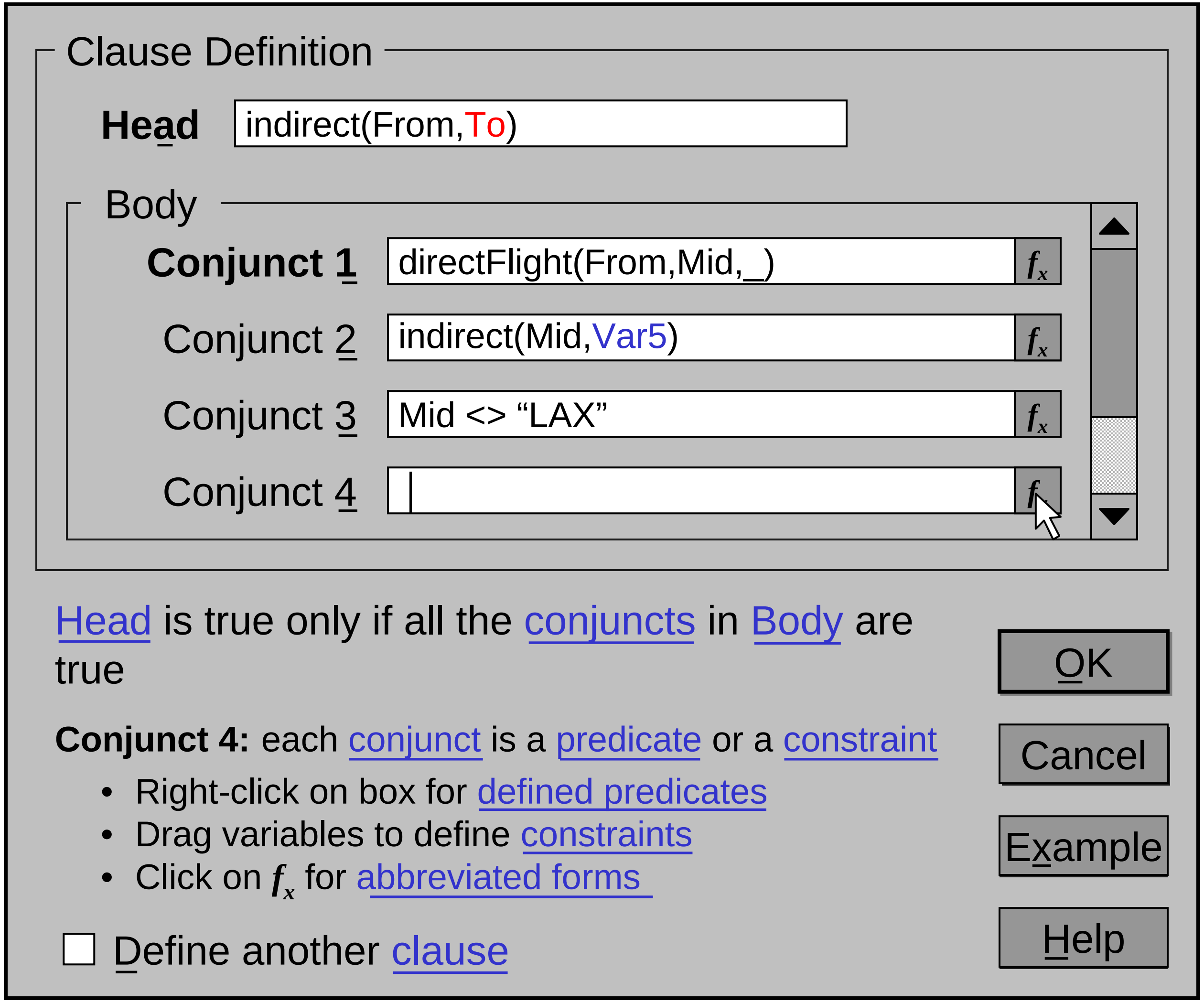
```
=indirect(From,To)
  IF      directFlight(From,To,_).
=indirect(From,To)
  IF      directFlight(Var1,Var2,Var3)
```

	A	B	C
1	directFlight		
2	From	To	D
3	ATL	SFO	
4	CDG	EWR	
5	CDG	VCE	
6	DCA	ATL	540
7	IAD	CDG	3850
8	IAD	LAX	2252
9	JFK	BWI	181
10	JFK	LAX	2437

- Identify variables by dragging
- Click constraints in

Wizard-Assisted Definition

- Enter formula in wizard
- Mouse assisted shortcuts available



Clause Definition

Head indirect(From,To)

Body

Conjunct 1 directFlight(From,Mid,_)

Conjunct 2 indirect(Mid,Var5)

Conjunct 3 Mid <> "LAX"

Conjunct 4 |

Head is true only if all the conjuncts in Body are true

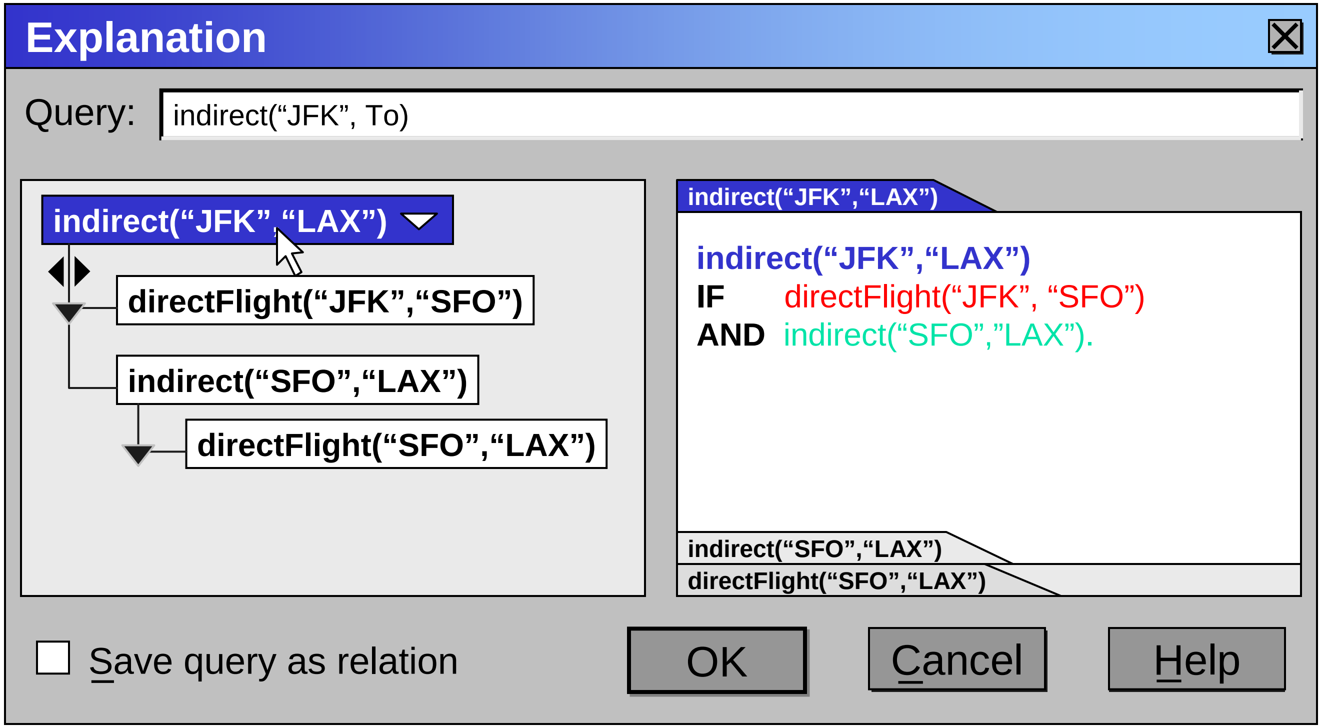
Conjunct 4: each conjunct is a predicate or a constraint

- Right-click on box for defined predicates
- Drag variables to define constraints
- Click on f_x for abbreviated forms

□ Define another clause

Explanation Facilities

- Invoked using right-click menu
- Displays proof tree
  - Color-coded feedback in spreadsheet
  - Browsable
- Allow entering arbitrary queries
- Allows saving result



Explanation

Query: indirect(“JFK”, To)

indirect(“JFK”,“LAX”)
- directFlight(“JFK”,“SFO”)
- indirect(“SFO”,“LAX”)
  - directFlight(“SFO”,“LAX”)

indirect(“JFK”,“LAX”)

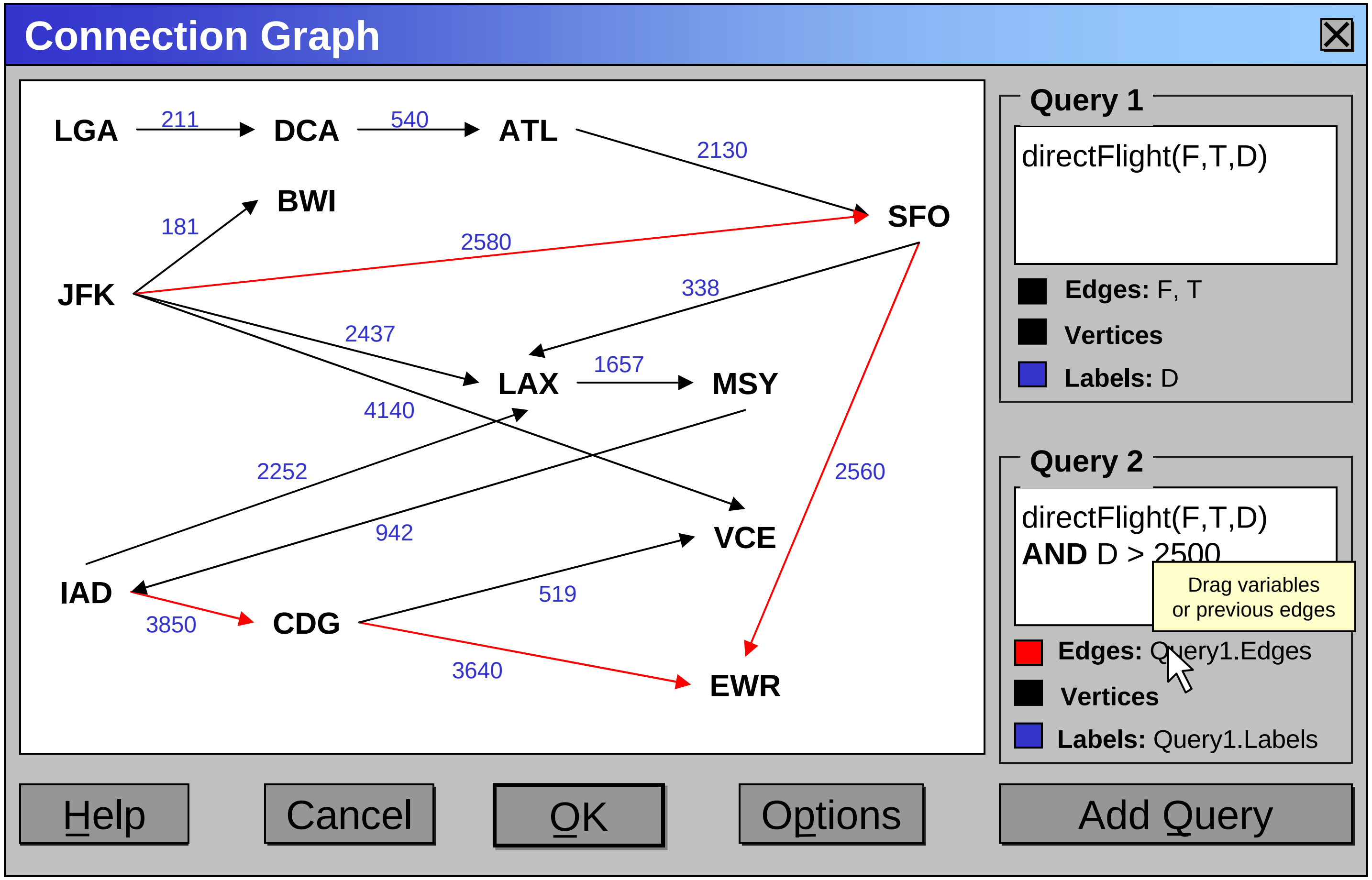
indirect(“JFK”,“LAX”)
IF directFlight(“JFK”, “SFO”)
AND indirect(“SFO”,”LAX”).

indirect(“SFO”,“LAX”)
directFlight(“SFO”,“LAX”)

☐ Save query as relation

Productivity Tools

Connection graph



Query 1

directFlight(F,T,D)

■ **Edges:** F, T

■ **Vertices**

🟦 **Labels:** D

Query 2

directFlight(F,T,D)
AND D > 2500

Drag variables or previous edges

🟥 **Edges:** Query1.Edges

■ **Vertices**

🟦 **Labels:** Query1.Labels

- More soon
  - Flow graph
  - …

Preliminary User Testing

- 8 volunteers
  - 3 advanced
  - 2 intermediate
  - 2 beginners — *NOT in target audience*
  - 1 theoretical computer scientist ...
- Outline of experiment
  1. Background questionnaire
  2. Illustration of Deductive Spreadsheet
  3. Walk through example and user interface

  Collected feedback at each stage

Feedback

- Advanced users
  - Followed example and suggested applications
  - General approval of user interface
  - Interested in all aspects of the Deductive Spreadsheet
  - Would use the Deductive Spreadsheet if it were available
- Intermediate users
  - Followed example and suggested applications
  - Disapproved of choice of some keywords in interface
  - Interest in many aspects of the Deductive Spreadsheet
- Beginners — *NOT in target audience*
  - Appreciated general objectives but difficulties with example
  - Trouble with wording of interface
  - Lot of interest in basic relational inference
    - Demanded simpler interface

Future Work

- Prototype
- Enhancements to User Interface
- Experimental assessment
  - User testing
  - Performance
  - Problem base
- Integration of other notions of “deductive”