Programming Rules using a Spreadsheet-based Interface

Gopal Gupta & Abhilash Tiwari

University of Texas at Dallas

Motivation & Objective

- Motivation
  - Non-experts should be able to program rules
  - Leverage existing technologies
    - Spreadsheets (most users familiar with them)
    - Existing reasoning engines (don't reinvent the wheel)
- Objective:
  - Develop a spreadsheet-based interface for programming constraints/rules for a class of apps.
  - Class of apps to focus: Solutions of CSPs

Constraint Satisfaction Problems (CSPs)

- Reasoning problems where the objective is to find a solution which satisfies a given set of constraints

  Solution == values for unknown variables

  CONSTRAINTS == RULES

- NP-complete: hard to solve, involve search
- Traditionally solved using search:
  - Constraint Propagation
  - Backtracking
  - Forward Checking
  - Local Search
  - Branch and Bound

Resource Allocation as a CSP

- Salient example of CSPs: Resource Allocation
- Why focus on resource allocation problems?
  - Ubiquitous
  - Complex
- Can be modeled as Constraint Satisfaction Problems (CSPs) over finite domain

  Recall CONSTRAINTS == RULES
- Fit into the spreadsheet paradigm

Examples of Resource Allocation Problems

- Time Tabling
- Scheduling
  - Flight Scheduling
  - Course Scheduling
  - Sports Events Scheduling
  - Employee Scheduling
- Degree Audits for University students

Spreadsheet Paradigm

- Spatial data-centric paradigm
- Tabular structure (consisting of rows and columns)
- Used for Manipulating Table(s) of Data
  - Data-Centric: User is always looking at the data; programming is done around the data
  - Data items in each row/column have similar characteristics

Spreadsheet Paradigm (Contd.)

- Programming done by replication
  - Replication is parameterized: give one example of a computation, then replicate it multiple times (with appropriate transformations applied).
  - No looping construct used: iterations replicated explicitly with the index variables set appropriately.
- Set of built-ins to perform familiar operations
  - E.g. Sum

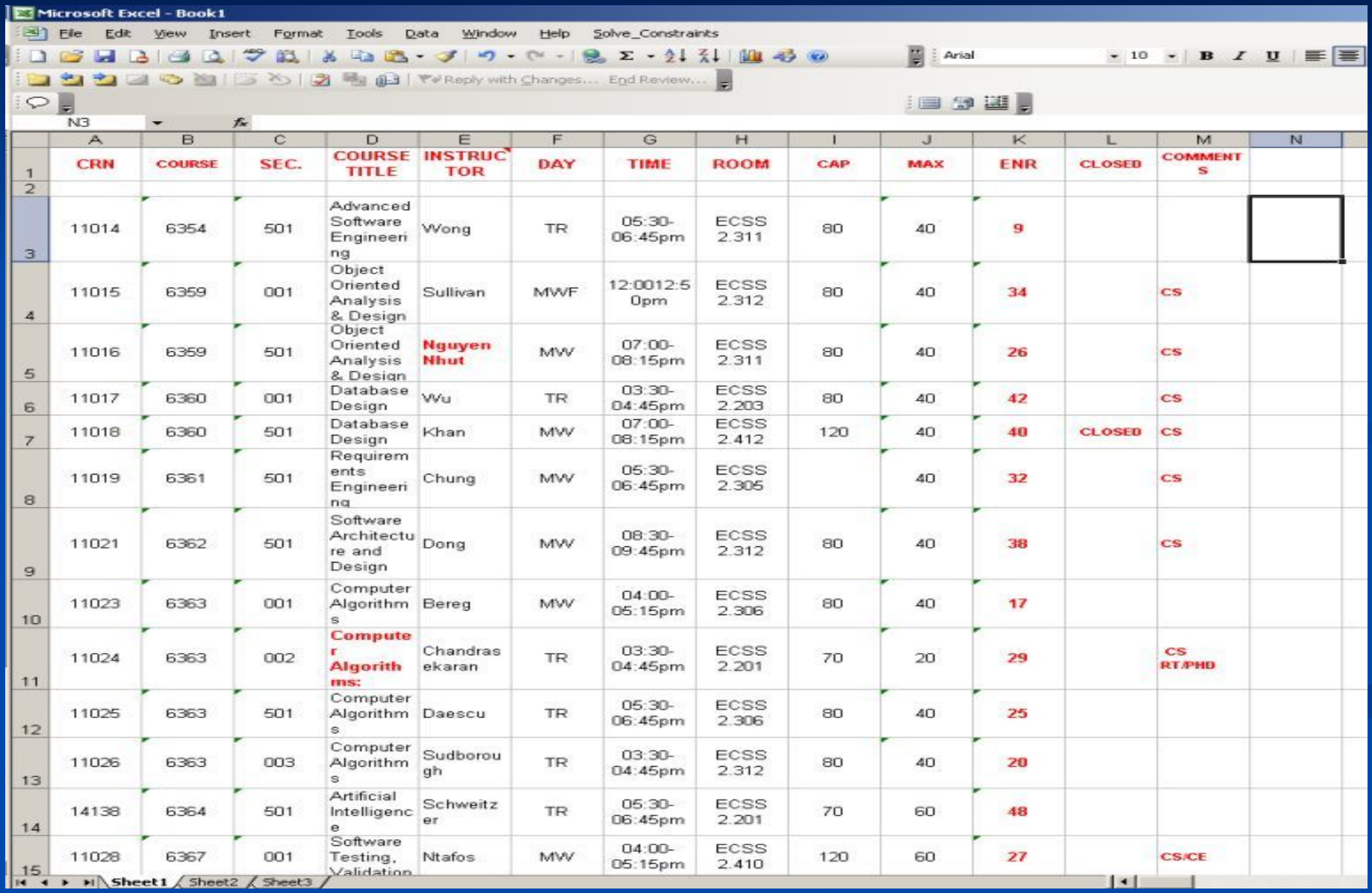
Spreadsheet Paradigm (Contd.)

- Familiar, interactive interface for handling multi-dimensional data
- Popular for arithmetic computations
- Current spreadsheets limited to arithmetic expressions
  - Arithmetic expressions are *interactively* entered and evaluated until desired results are obtained
  - Repetitive computations are performed by *copying expressions from one cell to a range of cells*, with appropriate transformation applied

Spreadsheets for Solving RAPs

- Resource Allocation Problems as CSPs
  - Most have a tabular, two dimensional structure
  - Constraints similar across rows and columns
- Due to this tabular nature, Spreadsheets can be used for modeling/solving RAPs
- provided we generalize functional arithmetic expressions to constraints (over finite domains)
- PlanEx: An intelligent spreadsheet based interface to model and solve CSPs

CS Class Schedule



Microsoft Excel - Book1

File Edit View Insert Format Tools Data Window Help Solve_Constraints

Arial 10 B I U

Reply with Changes... End Review...

N3

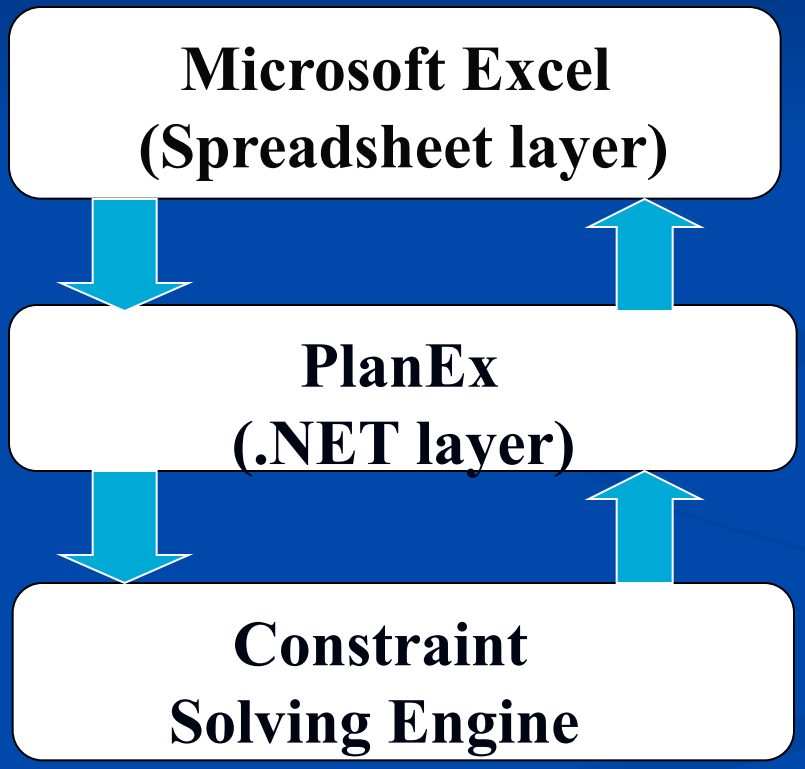
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	CRN	COURSE	SEC.	COURSE TITLE	INSTRUCTOR	DAY	TIME	ROOM	CAP	MAX	ENR	CLOSED	COMMENTS	
2														
3	11014	6354	501	Advanced Software Engineering	Wong	TR	05:30-06:45pm	ECSS 2.311	80	40	9			
4	11015	6359	001	Object Oriented Analysis & Design	Sullivan	MWF	12:0012:50pm	ECSS 2.312	80	40	34		CS	
5	11016	6359	501	Object Oriented Analysis & Design	Nguyen Nhut	MW	07:00-08:15pm	ECSS 2.311	80	40	26		CS	
6	11017	6360	001	Database Design	Wu	TR	03:30-04:45pm	ECSS 2.203	80	40	42		CS	
7	11018	6360	501	Database Design	Khan	MW	07:00-08:15pm	ECSS 2.412	120	40	40	CLOSED	CS	
8	11019	6361	501	Requirements Engineering	Chung	MW	05:30-06:45pm	ECSS 2.305		40	32		CS	
9	11021	6362	501	Software Architecture and Design	Dong	MW	08:30-09:45pm	ECSS 2.312	80	40	38		CS	
10	11023	6363	001	Computer Algorithms	Bereg	MW	04:00-05:15pm	ECSS 2.306	80	40	17			
11	11024	6363	002	Computer Algorithms:	Chandrasekaran	TR	03:30-04:45pm	ECSS 2.201	70	20	29		CS RT/PHD	
12	11025	6363	501	Computer Algorithms	Daescu	TR	05:30-06:45pm	ECSS 2.306	80	40	25			
13	11026	6363	003	Computer Algorithms	Sudborough	TR	03:30-04:45pm	ECSS 2.312	80	40	20			
14	14138	6364	501	Artificial Intelligence	Schweitzer	TR	05:30-06:45pm	ECSS 2.201	70	60	48			
15	11028	6367	001	Software Testing, Validation	Ntafos	MW	04:00-05:15pm	ECSS 2.410	120	60	27		CS/CE	

Sheet1 Sheet2 Sheet3

PlanEx

- PlanEx - Intelligent interface for solving CSPs
- Select and Click approach to constraint generation
- An Application level Add-In
- Implemented as a Visual Studio Tools for Office (VSTO) Add-In
- PlanEx = Microsoft Excel + .NET
  + Constraint Solving Engine

PlanEx Architecture



PlanEx Interface

- Microsoft Excel spreadsheet as Interface
  - Implemented as an Excel Add-In
- Cells can be thought of as variables
- Cell values can contain
  - A set of permissible values: e.g., 1..5
  - Constants: e.g., 10
  - Constraints: e.g., C2 #= C1+1

    e.g, COUNT (5, [B1,B3,B4], #=, 1)

PlanEx Features

- Constraints can be replicated over a range of cells replicated with appropriate transformations applied
- Constraint Generation - Select and Click style
  - Ex: Distinct, Frequency, Present_Once, If-Then
- Swap a compatible range of cells
  - Especially useful in Course Scheduling

Solution Generation

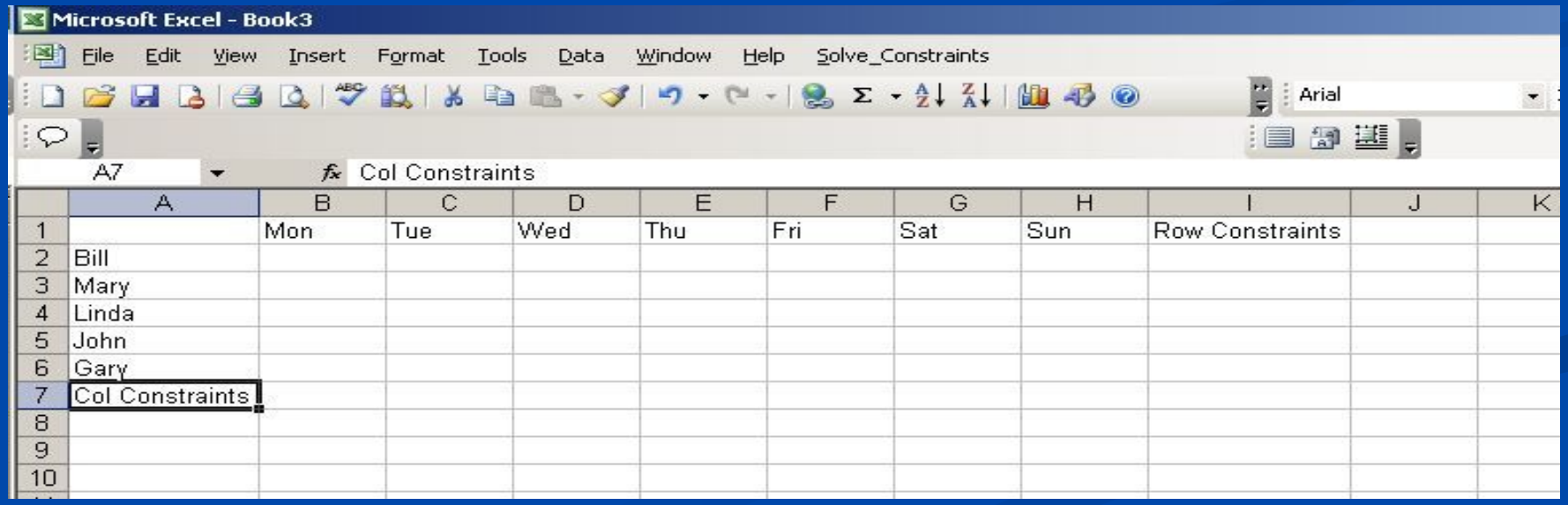
- Once all permissible values, constraints/rules are entered, and replicated, user generates solutions
  - Three clicks (in case there's *no domain mapping)*
- Once constraints/constants/permissible values are entered
  - the system automatically collects them
  - composes a CSP program
  - solves it using a back-end Engine based on Prolog
  - displays the solution

Example: Employee schedule

- Scheduling managers at a store:
  - Store hours : 8 AM to 11 PM, 7 days / week
  - Each manager must work 8.5 hrs / day (includes 0.5 hrs for lunch)
  - Each manager must work 5 days / week
  - At least one manager must be present at any moment
  - Managers working night shifts should not be allocated morning shift the following day
  - Schedule must be fair to all managers
- In most cases, this scheduling is done manually
  - Erroneous, leads to employee dissatisfaction

Solution: Employee Schedule

- Assume that there are 5 managers
- Each manager works 8.5 hrs per day either in
  - The morning shift (8:00 AM to 4:30 PM), or
  - The midday shift (10:00 AM to 6:30 PM), or
  - The evening shift (2:30 PM to 11:00 PM)



	A	B	C	D	E	F	G	H	I	J	K
1		Mon	Tue	Wed	Thu	Fri	Sat	Sun	Row Constraints		
2	Bill										
3	Mary										
4	Linda										
5	John										
6	Gary										
7	Col Constraints										
8											
9											
10											

An Empty Table

Solution: Employee Schedule (*continued*)

- Morning, midday and evening shifts are denoted by 5, 2 and 4 respectively
- 0 is used to indicate a manager's day off
- Domain of each cell: [0,2,4,5]
- User enters domain in one cell, copies it to the rest
- For no morning after night restriction, we enter the constraint:

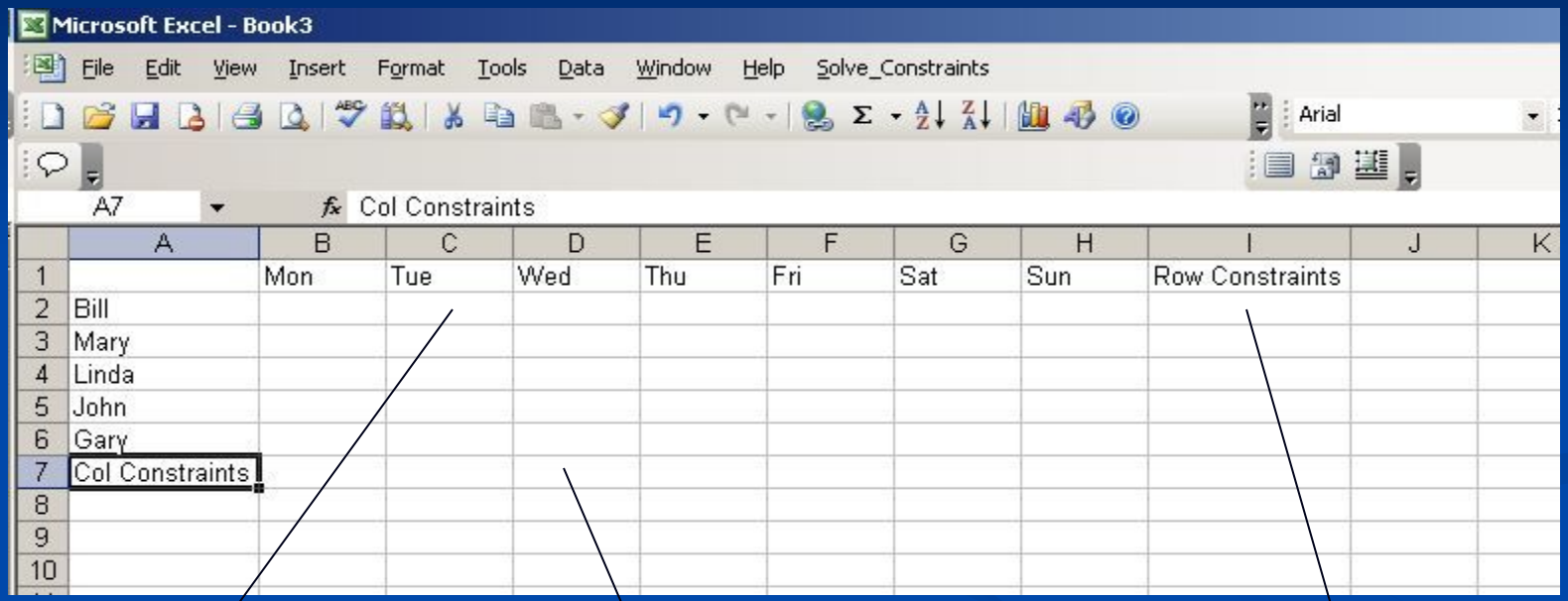
  C2 != B2 + 1 (copied everywhere)
- At least one manager is present at any time during the day:

  present_once(4,[D2,D3,D4,D5,D6]),
  present_once(5,[D2,D3,D4,D5,D6])
- No manager works for more than 5 days a week:

  count(0,[B2,C2,D2,E2,F2,G2,H2],=,2)
- Every manager has more or less same proportion of morning, midday and evening shifts:

  present_once([2,4,5],[B2,C2,D2,E2,F2,G2,H2])

Solution: Employee Schedule (*continued*)



	A	B	C	D	E	F	G	H	I	J	K
1		Mon	Tue	Wed	Thu	Fri	Sat	Sun	Row Constraints		
2	Bill										
3	Mary										
4	Linda										
5	John										
6	Gary										
7	Col Constraints										
8											
9											
10											

[0,2,4,5], C2 != B2 + 1

(Cell Constraints)

member(4,[D2,D3,D4,D5,D6]),
member(5,[D2,D3,D4,D5,D6])

(Column Constraints)

count(0,[B2,C2,D2,E2,F2,G2,H2],=,2),

sublist([2,4,5], [B2,C2,D2,E2,F2,G2,H2])

(Row Constraints)

Note: Cell constraints are replicated in all 35 cells, column constraints in B7 through H7 and row constraints in I2 through I6.

Solution: Employee Schedule (*continued*)

Microsoft Excel - Book3

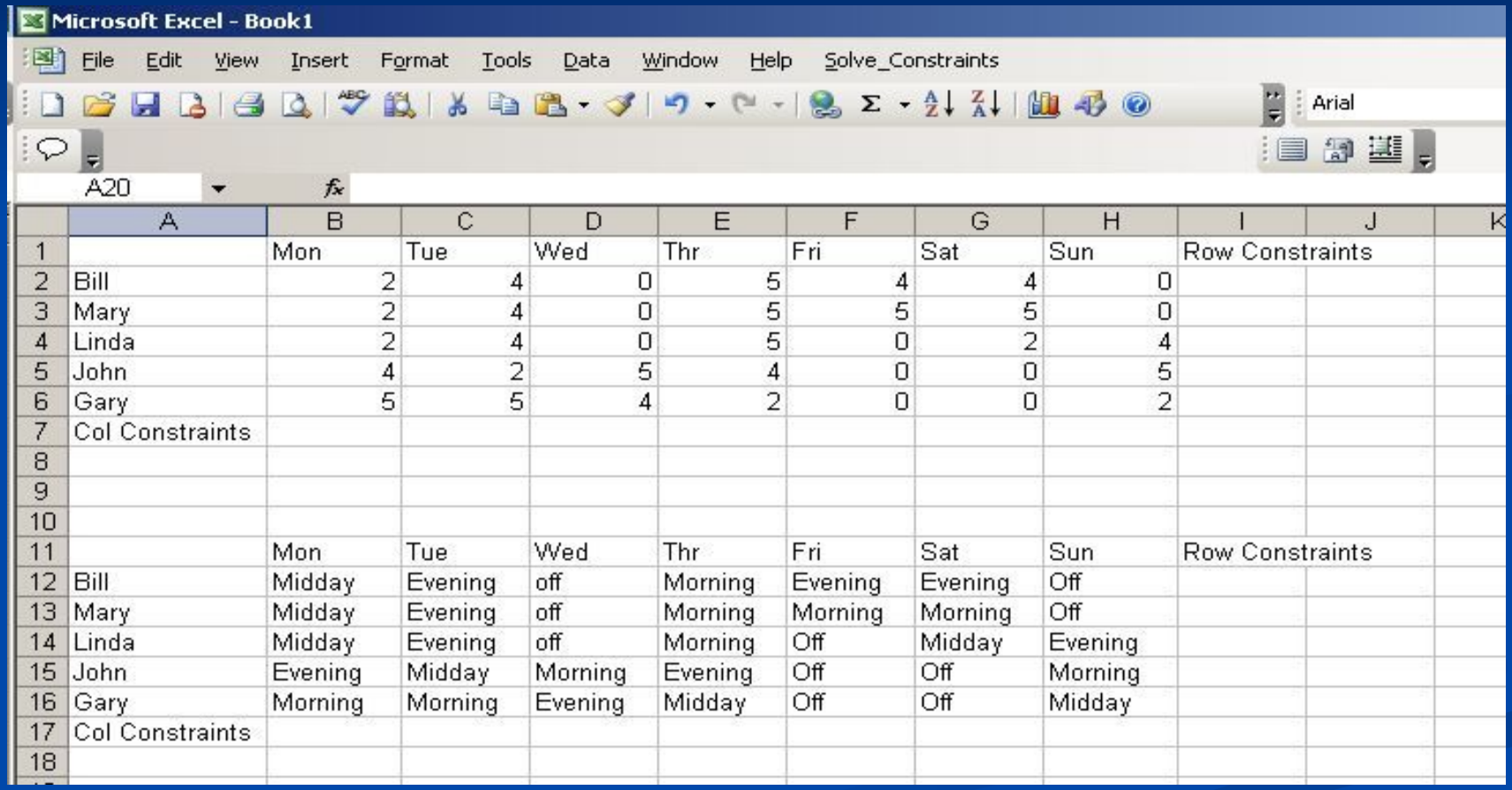
File Edit View Insert Format Tools Data Window Help Solve_Constraints

B2 = 2

	A	B	C	D	E	F	G	H	I	J	K
1		Mon	Tue	Wed	Thu	Fri	Sat	Sun	Row Constraints		
2	Bill	2	4	0	5	4	4	0			
3	Mary	2	4	0	5	5	5	0			
4	Linda	2	4	0	5	0	2	4			
5	John	4	2	5	4	0	0	5			
6	Gary	5	5	4	2	0	0	2			
7	Col Constraints										
8											
9											
10											

Displaying a solution

Solution: Employee Schedule (*continued*)



	A	B	C	D	E	F	G	H	I	J	K
1		Mon	Tue	Wed	Thr	Fri	Sat	Sun	Row Constraints		
2	Bill	2	4	0	5	4	4	0			
3	Mary	2	4	0	5	5	5	0			
4	Linda	2	4	0	5	0	2	4			
5	John	4	2	5	4	0	0	5			
6	Gary	5	5	4	2	0	0	2			
7	Col Constraints										
8											
9											
10											
11		Mon	Tue	Wed	Thr	Fri	Sat	Sun	Row Constraints		
12	Bill	Midday	Evening	off	Morning	Evening	Evening	Off			
13	Mary	Midday	Evening	off	Morning	Morning	Morning	Off			
14	Linda	Midday	Evening	off	Morning	Off	Midday	Evening			
15	John	Evening	Midday	Morning	Evening	Off	Off	Morning			
16	Gary	Morning	Morning	Evening	Midday	Off	Off	Midday			
17	Col Constraints										
18											

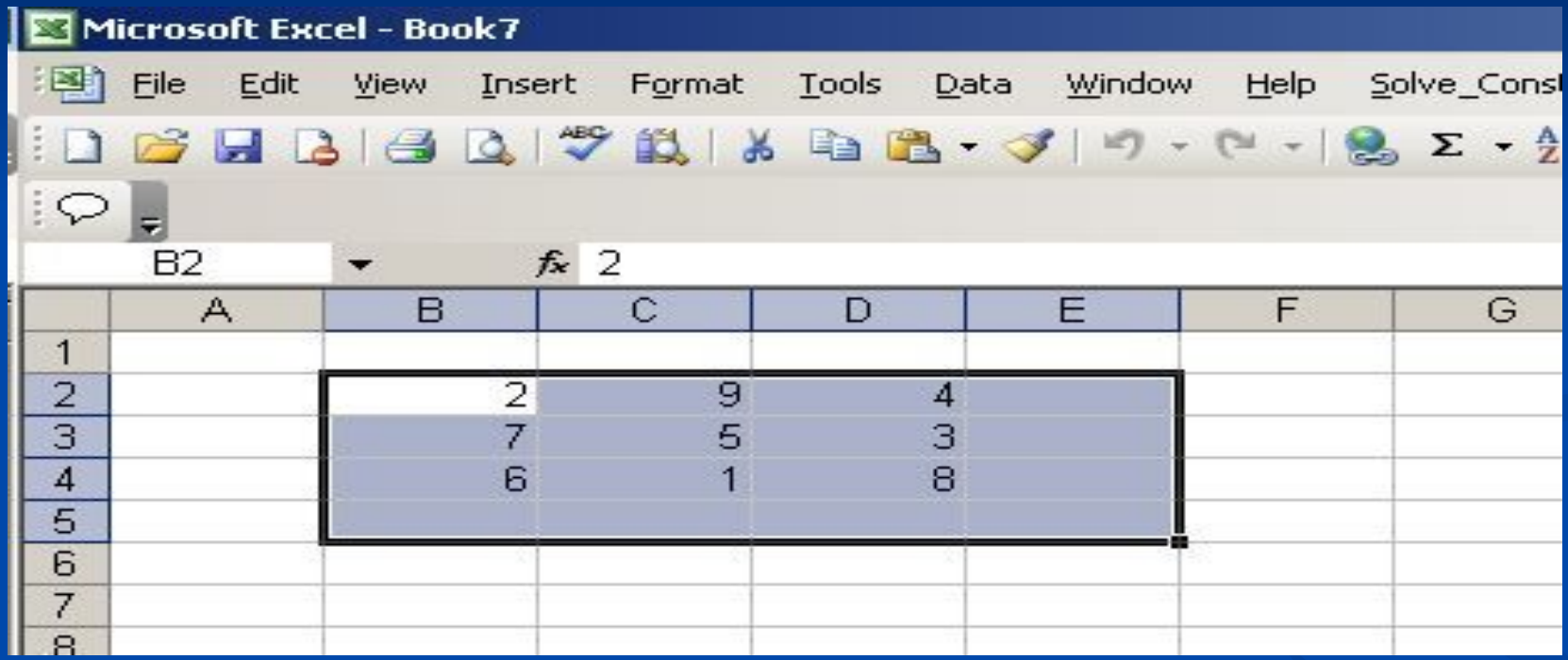
Displaying a solution along with a mapping of variable values

Example: The 3x3 Grid Puzzle

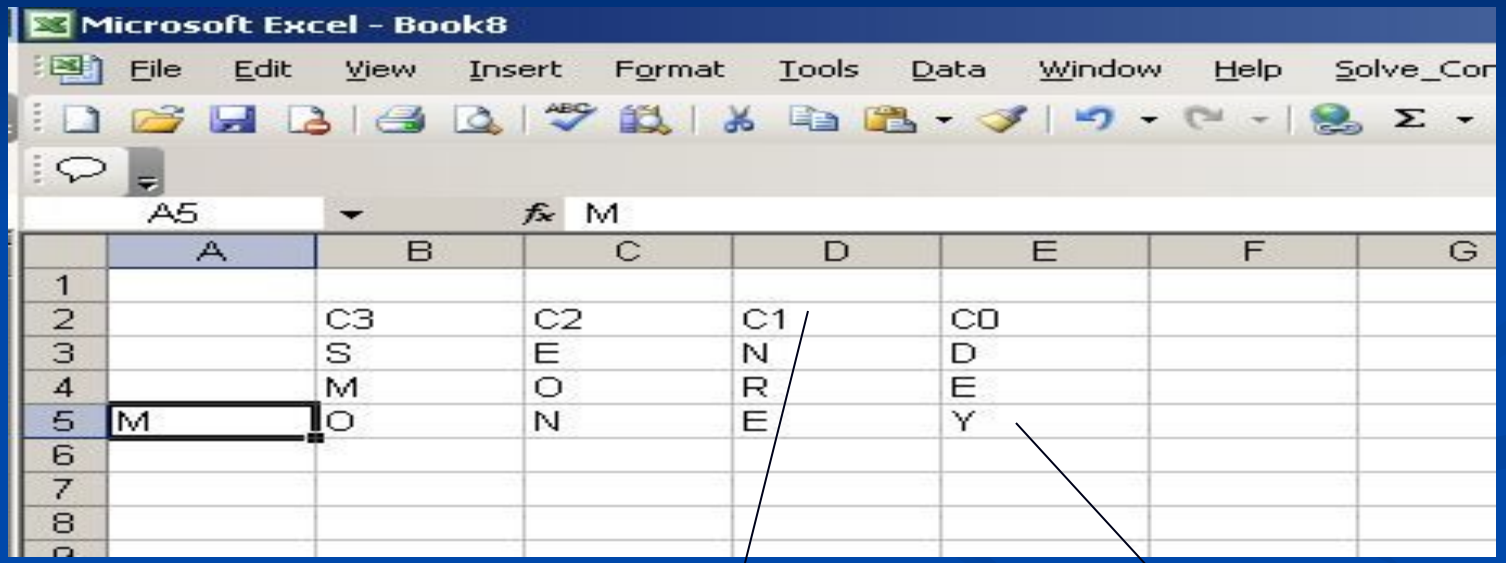
Cell constraints:

- B3: B3+C3+D3 #= 15, B3+B4+B5 #= 15
- C3: C3+C4+C5 #= 15
- D3: D3+D4+D5 #= 15
- B4: B4+C4+D4 #= 15
- B5: B5+C5+D5 #= 15, B5+C4+D3 #= 15
- D5: B3+C4+D5 #= 15, distinct ([B3,B4,B5,C3,C4,C5,D3,D4,D5])

Solution: The 3x3 Grid Puzzle



Example: Cryptarithmetic Puzzles



D2 #= (E2+E3+E4) DIV 10 E5 #= (E2+E3+E4) MOD 10

- Most puzzles have such a graphical structure; for example, Zebra puzzle
- PlanEx can be used for solving puzzles published in popular puzzle magazines

Conclusion

- Advantages of the PlanEx Approach:
  - Flexibility
  - Interactivity
  - Non-experts can use it (Expert System Shell)
    - Managers are resource allocators!!
    - Standard (deficient) spreadsheets currently used (sorting fn used a lot)
  - Domain specific knowledge can be incorporated
  - User and Constraint Engine cooperate to produce solutions
  - User can give partial solutions, the rest can be computed by PlanEx
- Disadvantages:
  - Works only for tabular CSPs
  - No automatic help if the system is over-constrained
- DEMO NEXT

Demo

Questions