Gambit Graphics User Interface
Version 0.96.3

An Interactive Extensive Form Game Program

This version by

Richard D. McKelvey
California Institute of Technology

Andrew McLennan
University of Minnesota

Theodore Turocy
Northwestern University

Part of the Gambit Project (p. 6)
California Institute of Technology
Funding provided by the National Science Foundation
Front end based on previous versions by Eugene Grayver
Contents

Introduction ........................................................................................................................................... 8

Installation and Support ....................................................................................................................... 9
Platforms .................................................................................................................................................. 9
Installation ................................................................................................................................................ 9
Technical Support and Bug Reports ....................................................................................................... 9

Representation of Games in GAMBIT ................................................................................................. 11
The Extensive Form .............................................................................................................................. 11
Labeling .................................................................................................................................................. 12
Numbering ............................................................................................................................................. 12
Strategies and The Normal Form .......................................................................................................... 12
Wildcard notation ................................................................................................................................... 13
Normal Form .......................................................................................................................................... 13
Types of Games ....................................................................................................................................... 14
perfect recall .......................................................................................................................................... 14
Games of incomplete information ........................................................................................................ 14

GAMBIT GUI ........................................................................................................................................... 16
File menu ................................................................................................................................................. 16
Data Types ............................................................................................................................................. 17
Dialogs ................................................................................................................................................... 18
Solutions Inspect .................................................................................................................................... 18
Normal Form Solution Inspect .............................................................................................................. 18
Extensive Form Solution Inspect ........................................................................................................... 19
Sorting and Filtering Solutions .............................................................................................................. 20
Adding and Editing Solutions .................................................................................................................. 21
Output Media Dialog ............................................................................................................................. 22
Standard Solution Dialog ...................................................................................................................... 23
Change Payoffs Dialog ........................................................................................................................... 23
Select Outcome Dialog ........................................................................................................................... 24
Draw Options Dialog .............................................................................................................................. 24
Inspect Node Dialog ............................................................................................................................... 25
Zoom Window ......................................................................................................................................... 26
Accelerators Dialog ............................................................................................................................... 26
Dominance Elimination Dialog ............................................................................................................. 27
Creating and Editing Supports ............................................................................................................... 28
Add Move Dialog ..................................................................................................................................... 28
Legends Dialog ....................................................................................................................................... 29
Table Window ........................................................................................................................... 30
Mathematical Errors .................................................................................................................. 30
Incorrect Solutions ...................................................................................................................... 31

Normal Form GUI ......................................................................................................................... 32
Normal Form Display .................................................................................................................... 32
Row and Column Players ............................................................................................................. 33
Strategy Profile ............................................................................................................................ 33
Normal Form Menu ....................................................................................................................... 33
File Menu (nfg) ............................................................................................................................. 33
Edit Menu (nfg) ............................................................................................................................. 34
Supports Menu (nfg) ...................................................................................................................... 34
Solve Menu (nfg) ............................................................................................................................ 35
View Menu (nfg) ............................................................................................................................ 36
Prefs Menu (nfg) ............................................................................................................................ 36
Normal Form Solutions .................................................................................................................. 37
NFG Standard Solutions ............................................................................................................ 38
NFG Custom Solutions ................................................................................................................ 38
Default Accelerator Keys ............................................................................................................... 39

Extensive Form GUI ...................................................................................................................... 40
Extensive Form Display .................................................................................................................. 40
Navigating the Extensive Form ...................................................................................................... 40
Drag and Drop ............................................................................................................................... 41
Extensive Form Menu ..................................................................................................................... 41
File Menu (efg) ............................................................................................................................. 42
Edit Menu (efg) ............................................................................................................................. 42
Subgames Menu (efg) ..................................................................................................................... 46
Supports Menu (efg) ...................................................................................................................... 46
Solve Menu (efg) ............................................................................................................................ 47
Inspect Menu (efg) .......................................................................................................................... 49
Prefs Menu (efg) ............................................................................................................................ 49
Extensive Form Solutions .............................................................................................................. 50
Subgames ........................................................................................................................................ 50
Extensive form Supports ............................................................................................................. 51
EFG Standard Solutions ............................................................................................................... 52
EFG Custom Solutions ................................................................................................................... 53
EFG Solutions and Subgames ....................................................................................................... 55
Default Accelerator Keys ................................................................................................................ 56

Solutions of Games ......................................................................................................................... 57
Strategy Profile .................................................................................................................. 72
Subgame.............................................................................................................................. 72
Subtree ................................................................................................................................. 72
Terminal Node ...................................................................................................................... 72
Topological Tree .................................................................................................................. 72
Copyright notice

Copyright (c) 2000, The Gambit Project, at California Institute of Technology and University of Minnesota.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice and this permission notice appear in all copies of this software and related documentation.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL THE GAMBIT PROJECT, THE CALIFORNIA INSTITUTE OF TECHNOLOGY, THE UNIVERSITY OF MINNESOTA, OR ANYONE ASSOCIATED WITH THE DEVELOPMENT OF COMPUTER SOFTWARE UNDER THE GAMBIT PROJECT, BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
Acknowledgments

The Gambit Project is a project for the development of computer code for the solution of extensive and normal form games. The software developed under the project is public domain. The Gambit Project is funded in part by National Science Foundation grants SBR-9308637 and SBR-9617854 to the California Institute of Technology and SBR-9308862 to the University of Minnesota.

The Gambit Project is an ongoing project, where the set of active participants has changed over time, and will be expected to change further in the future. The main contributors to this release are Richard D. McKelvey, Andrew McLennan and Ted L. Turocy. However, this release builds heavily on the previous releases, and we would be remiss not to acknowledge the significant contributions of those involved in those previous releases.

The Gambit project was begun in the mid 80's, at which time it consisted of an implementation of the GUI in Basic. The code has since been totally rewritten twice (first in C, then in C++), the functionality has been expanded, and the GCL has been developed as an alternative to the GUI to support econometric and computationally intensive use.

Richard McKelvey has directed the Gambit project since its inception in the mid 80's, and was one of the principal investigators on the NSF grants for the Gambit Project. He designed the original GUI, was involved in the design of the GCL (1993), and has been responsible for implementation of many of the algorithms in gambit.

Andrew McLennan has worked on the project since 1993, and was one of the principal investigators on the NSF grants for the Gambit project. Andy was involved in the original design of the GCL (1993). He has been responsible for the implementation of the polynomial solution code, which finds all roots of polynomial systems of equations, and (effective this release) is used as the default algorithm to find all Nash equilibria of n-person games, when n is greater than two.

Ted Turocy has worked on the project since 1992, working full time for a year and a half of that time. He has been the main programmer for the project, also responsible for supervising other programmers working on the project. Ted converted the code from C to C++, changed it to an object oriented style, wrote code for the container classes and for the current implementation of the extensive and normal form classes, and implemented the GCL. He also made substantial improvements to the GUI for this release.

In listing the contributors to previous releases, we must first mention Eugene Grayver and Gary Wu, each of whom worked for at least four years on the project, making major contributions:

Eugene Grayver worked on the project from 1994 through 1997. He wrote the code for the wxwin implementation of the GUI for versions .92 through .94, developed our Web page, handled the software distributions, and implemented the console interface for version .94 of the GCL.

Gary Wu worked from 1995 through 1998 on the project. He was responsible for large
portions of the GCL implementation, including the implementation of "listability", implementation of many of the built in functions, the command line editor, the original stack machine, and the Windows MFC interface for the GCL.

Numerous additional students at Caltech and the University of Minnesota have contributed to the Gambit Project:

Bruce Bell worked in 1989 on the first C version of the Gambit GUI, which was distributed in 1991-92 as version .13, and later (1994-1995) helped with the implementation of the tableau and LU decomposition code (used in the two person LCP and LP algorithms). Matthew Derer worked in the summer of 1993 to do an initial implementation of the GUI for XWindows. Anand Chelian worked summer of 1994 to do an initial implementation of the matrix and vector classes. Brian Trotter worked summer of 1994 on internals of the container classes and the extensive form classes. Todd Kaplan worked in the summer of 1994 and did initial work towards implementation of a multivariate polynomial class. Nelson Escobar worked over the summer of 1995 to improve the LU Decomposition code, and worked on a second implementation of the polynomial classes. Rob Weber did extensive testing from 1994 to 1996, and helped with the documentation of version .94. Geoff Matters worked summer of 1997 to make various improvements to the LP solver and on improvements to the built in function definitions in the GCL. Ben Freeman worked in the summer of 1998 to implement the integer tableau class, which is used to speed up the rational computations for two person games. Michael Vanier worked summer of 1998 through the spring of 1999 to set up the CVS repository system for source code control, work on the distribution scripts, and make various bug fixes to the GUI.

We are also indebted to Bernhard von Stengel at the London School of Economics for help and advice on implementation of the sequence form and for contributing his "clique" code for identification of equilibrium components in two person games.
Introduction

Gambit is a library of computer programs, written in C++, for building, analyzing, and solving n-person games, in either extensive or normal form.

The Gambit Graphics User Interface (Gambit GUI) is an interactive, menu driven program for accessing this library of programs. The Gambit GUI allows for the interactive building and solving of extensive and normal form games. It consists of two separate modules, one for extensive and the other for normal form games. In each module, you can build, save load and solve a game, and convert back and forth from one form of game to the other.

Despite its ease of use, the Gambit GUI is not suitable for repetitive or computer intensive operations. A separate program, the Gambit Command Language (GCL) is designed to be used for such operations on games. The GCL is a program which allows access to the functionality of the gambit library of programs through a high level programming language. The Gambit GUI and GCL are compatible with each other, in the sense that they each generate files that can be read by the other, and they call the same underlying library of functions.
Installation and Support

Platforms

The gambit GUI is developed and tested primarily on the following platforms:

- IBM PC and compatibles running MS Windows 95/98 or NT.
- Linux operating systems using Motif.

Executable code for the above platforms as well as certain other platforms is available at the Gambit web site (check the gambit web page for current information).

For platforms that are not currently supported, the gambit source code is available at our web site. The gambit GUI relies on the wxWin library, and versions could be compiled for any platform that is supported by wxWin. See the gambit and wxWin web sites for more details.

Installation

All of the gambit files can be found at the Gambit World Wide Web site at

http://hss.caltech.edu/gambit

You will need a Web browser such as Netscape to download the files. Follow the instructions there for installation.

Technical Support and Bug Reports

User feedback is an important part of Gambit's development cycle. Our design choices have been motivated in large part by our experiences and by some of the potential uses of the software we have imagined. But this is by no means complete, and so we are eager to hear from you, Gambit's users, to find out what features are particularly useful, and in what areas future releases might be improved.

The authors may be contacted via email at gambit@hss.caltech.edu. This address will forward a copy of your message to the development team. While a personal reply may not always be possible, we will read all correspondence and apply as many suggestions as possible to our future releases.

Although we have made every effort to ensure the reliability and correctness of the code, it is certain that errors of varying degrees of severity exist. If you experience problems with the software, send us email at gambit@hss.caltech.edu describing your problem.

Before reporting an incorrect solution, make sure and first read the section on Incorrect Solutions (p. 31).

When reporting a bug, it is important to include the following information:

- the version number
• the platform(s) on which the problem occurred

• as complete a description of the circumstances of the problem as possible, including a sequence of steps which reproduces the problem. Without this information, it will be difficult for us to identify the source of the problem to correct it.

At this time, no formal technical support mechanism exists for Gambit. As time permits, we will make every effort to answer any and all questions pertaining to the program and its documentation.

We hope you will find Gambit a useful tool for research and instruction.
Representation of Games in GAMBIT

This chapter describes some notation and concepts from game theory that are necessary to understand how Gambit represents and solves games. This chapter assumes a basic understanding of game theory. Definitions of the main terms are given intuitively in the Glossary.

The Extensive Form

The extensive form is a detailed description of a sequence of decisions that must be made by a group of individuals. To illustrate the basic ideas, Figure 1 shows how the extensive form of a simple two player poker game (from [1] is represented in the Gambit GUI.

![Figure 1: GAMBIT Display of Extensive Form of a Simple Two Player Poker Game](image)

In Gambit, the extensive form game tree, is drawn with the root node furthest to the left, branches going from left to right, and successive nodes to the right of their predecessors. So nodes further to the right represent decisions that occur later in time. Each player, (including chance) is represented by a color.

A node is represented as a horizontal line. Every non-terminal node is a decision node and is represented by the color of the player who makes a decision at that node. Nodes are connected by branches, which are represented by two connected line segments, the "fork" and the "tine". The fork is used in the graphics display to indicate graphically the probabilities of taking each branch. The tine is used as a location at which to display textual information about the branch. In gambit Outcomes can be attached to either non terminal or terminal nodes and can be displayed either by label or by payoff next to the node to which they are attached.

Information sets in Gambit are identified by a pair of numbers, the first representing the player number and the second representing the information set number. This information can be displayed on the extensive form next to each node to identify which nodes are in which information sets. Whenever possible information sets are also
represented by vertical lines connecting the nodes that are members of the same information set. It is only possible to do this when nodes in the same information set are at the same level of the tree. When two member nodes of the same information set are not at the same level, the information set is indicated by a "lagged" line, the end of which points in the direction of the next node, as illustrated in Figure 2.

Figure 2: GAMBIT Display of information set when member nodes are at different levels

Labeling

There are three locations at which each node can be labeled, (above, below, and to the right) and two at which each branch can be labeled (above and below.) You can choose which information to display in the various positions by selecting the information in the Prefs->Legend (p. 49) menu.

Numbering

In order to associate strategy profiles in the normal form with the corresponding behavioral strategies in the extensive form, (and vice versa) you will need to understand how the branches and information sets are numbered. The numbering of branches and information sets in the extensive form is used to number pure strategies in the normal form.

The numbering of the branches is always from top to bottom. Thus, the highest branch at any node is always branch 1, the next branch is branch 2, etc. An action consists of the set of branches in an information set with the same branch number.

In Gambit, each information set has a unique information set ID, consisting of the player followed by the information set number. Information sets for each player are numbered consecutively, in indexed traversal order. The information set ID can be displayed at the decision nodes by selecting it in the Prefs->Legend (p. 49) menu.

Strategies and The Normal Form
A pure strategy for a player is a function that associates with each of its information sets, a chosen action at that information set. A pure strategy n-tuple is a vector of strategies, one for each player. The normal form for a game is a mapping which associates with each pure strategy n-tuple, an expected payoff to each player from playing that strategy.

In the extensive form of the game in GAMBIT, the information sets for a player are numbered from 1 to J, where J is the number of information sets that player i has. We denote the pure strategy in which player i adopts strategy a in its first information set, b in its second, and c in its third abc. So if player i has three information sets, where the first two information sets contain three branches and the last contains two branches, then player i has a total of 18 strategies, and 312 indicates the strategy where the player chooses the third branch at its first information set, the first branch at its second information set, and the second branch at its third information set.

In the extensive form of Figure 1 (p. 11), Player 1 has two information sets, each with two choices, so it has a total of four strategies. They are 11, 12, 21 and 22. Player 2 has one information set, with two branches, so they are labeled 1 and 2. The strategy 21 for player 1 represents the strategy of choosing FOLD with a Red card, and RAISE with a Black card.

**Wildcard notation**

To represent a collection of strategies which differ only in the choice at a particular information set, we use a "wildcard" notation. For example, if a player has three choices at its second information set, then the notation 3*2 is used to represent the collection of strategies, \{312, 322, 332\}.

The * notation in GAMBIT serves the same purpose as the "wildcard" character in DOS file specifications. To get a good appreciation for how the wildcard works, you should exit GAMBIT and WINDOWS, go to the root directory of your hard drive, and type `del *.*[RET]`.

**Normal Form**

The normal form of a game is a mapping which associates with each strategy profile a vector of payoffs for the players. The normal form will thus be an n dimensional matrix, with each cell containing a vector of length n.

Every extensive form game has an associated normal form. The payoff for a given player under a particular strategy is computed as the sum of the realization probabilities of each node times the value to the player of any outcome at that node. The payoffs in the normal form are simply the expected payoff to each player from the given strategy profile.
Figure 3 gives the normal form for the extensive form game of poker illustrated in Figure 1 (p. 11).

In the game of Figure 1 (p. 11), with the strategy profile \((12, 1)\), the realization probability of the terminal node on the path RED, RAISE, MEET, with a payoff of \((2.00, -2.00)\) is \(1/2\), and the realization probability of terminal node on the path BLACK, FOLD, with a payoff of \((-1.00, 1.00)\) is \(1/2\). All other terminal nodes have realization probability of 0 at this strategy profile. Taking expected values, this gives a payoff of \((0.50, -0.50)\), which is the entry in the normal form for this cell.

**Types of Games**

**perfect recall**

A game of perfect recall is a game in which individuals do not forget what they have known or done earlier in the game (see, eg., [1]). This requires that information sets must refine earlier information sets, and if there is an information set, one of whose nodes follows a choice by the same player at a previous information set, then all of the members of that information set must follow the same choice.

GAMBIT does not enforce perfect recall on the extensive form games that you build. However, certain algorithms in GAMBIT only work on games of perfect recall. To find out if a game is perfect recall, you can go to the Inspect->GameInfo menu. GAMBIT will allow you to convert a game without perfect recall to normal form and solve for optimal mixed strategies. The problem comes in converting the mixed strategy back to a behavioral strategy. If the game has perfect recall, then by Kuhn's theorem (see eg., [2]), any mixed strategy can be converted back to a behavioral strategy which is realization equivalent. If the game does not have perfect recall, this is not always possible. In this case, attempting to convert a mixed to a behavior strategy may yield unpredictable results which do not correspond to a valid behavior strategy.

**Games of incomplete information**

Games of incomplete information are games in which different players have different beliefs about the underlying parameters (such as the utility functions, strategy sets, or
number of players) of the game. The standard way of treating such games is given in [3], where it is shown that such games are equivalent to games in which players have some common prior distribution over the joint distribution of characteristics, and then individuals observe their own type, but not that of the other players in the game.

Games of incomplete information can be modeled in GAMBIT by having an initial chance move determine the distribution of types, and then defining information sets so each individual observes only their own type.
GAMBIT GUI

The Gambit GUI consists of two self-contained modules for representing, displaying and manipulating games. A game can be viewed in either the extensive or normal forms. Although independent, the modules are integrated and it is possible to solve the game in the normal form while viewing it in the extensive. It is also possible to go from the extensive form representation to the normal form (but not vice versa at the present).

When you first launch the gambit GUI, you will see a screen as in Figure 4

![Gambit GUI Introductory window](image)

**Figure 4: Gambit GUI Introductory window**

You can select the module to be used in the File menu, in one of two ways:

- Build a new game by selecting **File->New**, and then choosing either Extensive or Normal. The toolbar can be used to speed up frequently used commands.

- Load an existing game by selecting **File->Open**, and then selecting a file that contains a previously saved game in either extensive or normal form.

**File menu**

The **File** menu is used to launch gambit, by either loading a new game or loading a previously saved game.

- **New**: Used to create a new normal or extensive form game.
  
  **Normal**: You must input the number of players and the number of strategies for each player, and then the **Normal Form GUI** (p. 32) is started with a default normal for game of that dimension. The default game has no outcomes defined. So all payoffs are zero.

  **Extensive**: You must select the number of players (default is two), and then the **Extensive Form GUI** (p. 40) is started with a default game, which consists of the trivial game with only one node (the root node) and no actions for the players. The **Edit Menu** (p. 42) in the extensive form GUI is then used to build up the desired game.

- **Open**: You will get a standard File Selector dialog box from which you can either directly enter the name of the file to open, or browse the directory system for the...
desired file. Files containing extensive form games are given the default extension of .efg in Gambit, and files containing normal form games are given the default extension .nfg. If you select a normal form file, you will bring up the Normal Form GUI (p. 32) module. If you select an extensive form file, you will bring up the Extensive Form GUI (p. 40). When you save an extensive or normal form game in Gambit, it is written in a standard format. Note that the file format is forward, but not backward compatible: Previous versions of Gambit may not be able to read files written by this version, but this version should be able to read files written by previous versions of Gambit.

**Quit:** Quits the Gambit GUI, and closes all child windows.

**Data Types**

Gambit can do computations in either floating point (float) or exact arithmetic (rational). Each number in the extensive or normal form game has an internal representation, or "precision" which is either floating point or rational.

**Float:** Floating point numbers are represented internally as double precision floating point numbers, which on most machines results in about 15 digits of accuracy. The benefit of floating point calculations is speed, since floating point numbers are designed to fit in a fixed amount of storage, and arithmetic operations are coded in hardware. However, in some games, floating point calculations can result in cumulation of roundoff errors that will lead to either incorrect solutions or failure to find solutions that exist.

**Rational:** Rational numbers are represented internally as the quotient of two arbitrary precision integers. Doing calculations in rationals will guarantee that the answer is exact for those algorithms that support rational calculations. However calculations in rationals are slower than floating point calculations by one or two orders of magnitude. Currently the only algorithms supporting calculations in rationals are the following two person algorithms: LpSolve, LcpSolve, and EnumPure and EnumMixed.

The precision of a number is indicated in output by the way it is printed. If the number contains a decimal point, then its internal representation is floating point. If it has a / or is written as an integer (with no decimal or /), then it is a rational. Similarly, when you enter a number, the internal representation is determined by the way it is entered.

The data for a game can be a mixture of floating point and rational numbers. Any preliminary calculations in Gambit (such as elimination of dominated strategies, conversion to normal form) are done in the best precision that is supported by the data of the game. For example if a calculation involves a floating point and rational number, the result will be floating point.

Some of the algorithms in Gambit support either floating point or rational precision. The data for the game will be converted to that precision, and the resulting solutions returned in that precision. In some cases, conversion from rational to floating point can lose precision. For example, the number 1/3 does not have an exact representation in floating point. Thus, in cases where exact solutions are required, it is good advice to
enter all data in rational precision.

Dialogs

Solutions Inspect

The solution inspection dialog displays information about Mixed or Behavior Profiles that are either returned by algorithms or entered by the user. The dialogs for the Normal Form (p. 18) and Extensive Form (p. 19) solutions have many features in common, and hence are described together.

The number of the solution that is currently selected will be highlighted in the first column. The currently selected solution will also be displayed in the corresponding Extensive or Normal form window. To change the current solution (and thus change the display in the corresponding Extensive or Normal form window), double click (control-click in Linux) on the # of the desired solution. To quickly browse the solutions, the Update Solutions Dynamically option may be used. With this option on, the solution will change automatically once the cursor moves to a different solution #. Double clicking on the first row deselects the current solution.

Note that if the entries in the extensive or normal form are changed, then any solutions to the game are deleted, and any link to an associated extensive form game (if it existed) is broken.

A variety of data dependent on the generating algorithm are available for most Mixed or Behavior profiles. Not every category is relevant for every profile. If a certain data member is either not relevant or not available, it will be displayed as ----. The display of each data member is toggled in the Options dialog. Note that the ID data member is always displayed (it is an increasing (not necessarily consecutive) index used to identify each solution).

Solutions may be deleted by pressing the Delete button while the solution is selected. A solution may be edited by pressing the Edit (p. 21) button, or a new one created with the Add (p. 21) button. Note that an edited or a newly created solution will have User as the Creator and will have most data members undefined. Double clicking on any of the strategy values will automatically call up the solution edit dialog. To facilitate working with a large number of solutions, the Sort/Filter (p. 20) functionality is available.

Normal Form Solution Inspect

This dialog displays information about Mixed Strategy Profiles that are either returned by algorithms or entered by the user. This dialog is accessed through the View->Solution selection in the normal form menu.
The probabilities of each strategy are arranged with each player on a separate row, and strategies in consecutive columns. Each cell consists of the strategy number (or name), followed by a colon, and followed by the probability of that strategy. If the display zero prob option is not turned on, strategies with zero probability will not be displayed.

The Extensive and Normal form Solution Inspect dialogs have many features in common. See the Solution Inspect (p. 18) for a description of these common features.

If the underlying Normal Form game was generated from an Extensive Form game, and if the link between the two is still valid, the Mixed->Behav button will be enabled. To project the solution to the extensive form, and immediately display it there, press this button.

Note: If you get a solution that looks incorrect, for example with probabilities that add to more than one, see the section on Incorrect Solutions (p. 31).

Extensive Form Solution Inspect

This dialog displays information about Behavior Strategy Profiles that are either returned by algorithms or entered by the user. This dialog is accessed through the Inspect->Solution selection in the extensive form menu.
The probability data for each information set is displayed on a separate line. The Iset column gives the info set ID in the form of (player #,iset #). The next cell contains a vector of probabilities in which each entry corresponds to an action that can be taken at that info set. If a player has more than one info set, additional lines are used. This format is repeated for each player in the game.

The Extensive and Normal form Solution Inspect dialogs have many features in common. See the Solution Inspect (p. 18) for a description of these common features.

A useful feature when navigating large extensive form games is Hilight Infosets. It is toggled on or off from the EF window Inspect -> Infosets. When enabled, double clicking on an info set in the EF window will highlight the corresponding info set in the Solution inspect window. Double clicking on an info set (Iset column) in the Solution inspection window will highlight the corresponding info set in the EF window.

For extensive form games, an additional level of solution inspection is provided by the Node Inspect window. The window may be turned on/off through the Inspect->Info menu. When enabled, a small window displays solution information pertaining to the cursor node. The window can be dismissed by turning the option off in the same menu.

For configuration and output features of this dialog see the generic Table Window description.

Note: If you get a solution that looks incorrect, for example with probabilities that add to more than one, see the section on Incorrect Solutions (p. 31).

Sorting and Filtering Solutions
Figure 7: Sort/Filter Dialog Box

To facilitate work with a large number of solutions, Gambit implements solution sorting and filtering. These features are accessed by the Sort/Filter button in the solution inspect window. Clicking an item in the Sort By box will sort the solutions by their values on that item. [Double clicking on the column heading in the first row of the Solution Inspect window will also sort the solutions by the corresponding column]. Deselecting an item in the Filter By box will cause the solutions that satisfy that item not to be displayed. (I.e. to filter out all NASH equilibria, deselect the Yes item in the Nash listbox; to display just solutions generated by the QRE algorithm, deselect all items except for QRE in the Creator listbox. Note that the filter option may lead to some confusion if newly generated solutions are automatically filtered out.

Adding and Editing Solutions

Figure 8: Edit Mixed Solution Dialog Box
Gambit allows the user to add a new mixed or behavior profile or edit an existing solution for the purpose of investigating its properties. Note that profiles that are added or edited do not have to have probabilities that add to one.

These features are accessed via the Add and Edit buttons in the solution inspect window. The probabilities for each strategy can be edited by moving the cursor to the appropriate cell in the Edit Mixed Solution dialog box. Selecting OK will cause the profile to be added to the list of solutions in the Solution Inspect window. Any profile that is added or edited will appear in the Solution Inspect Window with a creator of User. When a solution is edited, typically its properties will change.

**Output Media Dialog**

Many of the windows in Gambit have a menu item which can be selected to generate output to a printer or a file. When you select this option, you will get a dialog box which will ask the medium to which the output is to be generated.

![Output Media Dialog](image)

**Figure 9: Output Media dialog**

You can select one of the following choices:

- **Printer:** This will send the output directly to your printer.

- **Postscript:** Writes a postscript file, which can be viewed by a postscript viewer (such as ghostscript) or sent to a printer supporting postscript.

- **Clipboard:** Writes to clipboard

- **MetaFile:** Writes to metafile.

- **PrintPreview:** This will generate a display on your screen which will show the positioning of the output on the page.

In some cases, there will be a toggle fit to page. If you select this option, then the entire output of the window will be displayed, even if it is not all visible on your screen. If you do not select this option, then only the portion that is visible on your screen is output.
Note: When using the postscript option, on some platforms the only a portion of the file may be displayed when viewing the file under ghostscript. We do not know the reason for this. However, the file will display correctly when incorporated into latex documents via psfig, for example.

Standard Solution Dialog

This dialog uses standard settings to compute various types of equilibria. For more information on the settings used, see the section on NFG Standard Solutions (p. 38) for the normal form or EFG Standard Solutions (p. 52) for the extensive form.

![Standard Solution dialog for normal form games](image1)

Figure 10: Standard Solution dialog for normal form games

![Standard Solution dialog for extensive form games](image2)

Figure 11: Standard Solution dialog for extensive form games

Select one choice in each box. Not all combinations are available for all games, in which case, some choices will be greyed out. For example Rational computations are only supported on certain algorithms on two person games, Perfect equilibria can only be computed for two person games.

Change Payoffs Dialog

This dialog used to enter or edit the payoffs of an outcome in the extensive or normal form. It is accessible through the Edit->Outcome->New or Edit->Outcome->Payoffs.
menu selections in either the Normal or Extensive Form GUI. It can also be accessed by double clicking on either a cell of the normal form, or an outcome in the extensive form.

Figure 12: Change Payoffs dialog

Select Outcome Dialog

This dialog used to select an outcome in either the extensive or normal form. It is accessible through the Edit->Outcome->Delete or Edit->Outcome->Attach menu selections in either the Normal or Extensive Form GUI.

Figure 13: Select Outcome dialog

Draw Options Dialog

This dialog used to select parameters affecting the layot of the game tree. It is accessible through the Prefs->Display->Layout menu selection in either the Extensive Form GUI.
All lengths are in pixels. The **Show Infoset Lines** choicebox is used to select a mode for display of information sets on the extensive form. **None** means no lines are drawn connecting members of information sets. **Same Level** means only members of an information set at the same level of the tree are connected with lines. **All Levels** means that all nodes are connected, using lagged lines. See Figure 2 in the *Extensive Form* (p. 11) section for an illustration.

**Inspect Node Dialog**

This dialog used to display solution information about the currently selected node in an extensive form game. The information is continually updated as the game is navigated. This dialog is accessible through the **Inspect->Cursor** menu item.

---

Figure 14: Draw Options dialog

Figure 15: Inspect Node dialog
The information to display can be toggled via the **Display** menu item.

**Zoom Window**

The zoom window is used to provide a magnified window into the selected node of the extensive form game. This dialog is accessible through the **Inspect -> ZoomWindow** menu item.

![Zoom Window](image)

**Figure 16: Zoom Window display**

The display is centered around the currently selected node in an extensive form game. The magnification is controlled by the + and − toolbar buttons on the Extensive Form display. The information in this dialog is continually updated as the game is navigated.

**Accelerators Dialog**

Many frequently executed commands are much more efficiently entered through the keyboard than by using a mouse. To speed up the use of the GUI for an experienced user, most of the ‘mouseable’ commands can also be done by entering a combination of keys on the keyboard. Both the normal and the extensive forms possess this functionality. The accelerator key setting is accessed through the **Prefs -> Accel** menu item.
Figure 17: Accelerators dialog

In this dialog, the event to be changed or defined is first selected from the choicebox on the left. If this event is associated with a key combination already, the key will be displayed in the box on the left, otherwise, the entries in the box will be blank. A new key combo can be assigned by selecting the combo in the box and pressing Set. The association can be removed by selecting None.

GAMBIT comes with a pre-defined set of accelerator keys described in the corresponding extensive (p. 56) and normal (p. 39) sections. Any of these default command-key associations can be edited or removed.

Dominance Elimination Dialog

This dialog is used to compute undominated strategies for a normal or extensive form games, and return a new support consisting of undominated strategies. It is accessed via the Supports->UnDominated menu item.

Figure 18: Dominance elimination dialog

The following parameters can be selected:

Eliminate Iteratively: Select this checkbox to iteratively eliminate dominated strategies. The result will be a nested sequence of supports, with the last one consisting of strategies that are undominated in the final support.

Type: You can select the type of domination to use: weak or strong domination.

Method: You can select whether to eliminate only strategies that are dominated by a single (pure) strategy or by a combination of some strategies (mixed). Elimination by mixed strategies is only available for normal form games. For extensive form games this choicebox will also allow selection of Conditional
domination, which means that dominance computations are done using payoffs that are conditional on reaching the information set of the strategy, rather than on the payoff at the root node.

**Precision:** This option is only available for mixed elimination, and determines whether the resulting computations are done in Rational or Floating point.

**Players:** You may select which players to eliminate dominated strategies for. The default is all players, meaning that dominated strategies for each player are computed over the initial support, and then eliminated simultaneously to create the new support.

### Creating and Editing Supports

This dialog is used to create a new support or edit an existing support. It is accessed through the **Supports->New** or **Supports->Edit** menu item. Only the normal form version is shown, as the functionality for the extensive form is similar.

![Support Selection dialog](image)

**Figure 19: Support Selection dialog**

The highlighted strategies are those that are included in the support. A new support starts with the default of the *full support*, editing a support starts with the selected support as the default. To deselect a player’s strategy, click on it in the appropriate listbox. The strategy will no longer be highlighted. When the dialog is dismissed, the newly created support will be added to the support list.

### Add Move Dialog

This dialog is used to add a move to the extensive form game, and is accessed through the **Edit->Node->AddMove** or **Edit->Node->InsertMove** menu items in the extensive form GUI.
Figure 20: Add Move dialog

Select the player to be in control of the move by clicking on a player in the Player choicebox. The player can be either chance or an existing player, or a "New Player." If New Player is chosen, a new player will be created and given a default name "Player #." If this node is to belong to an existing information set, choose the desired infoset from the Infoset choicebox. If a new information set is to be created, just enter the number of branches desired. If the node created was a CHANCE node, you will be prompted for the probabilities associated with each branch.

Legends Dialog

This dialog is accessed through the Prefs->Legends menu item, or by double clicking (or Ctrl-RightClick) the right mouse button on the extensive form display.

Figure 21: Legend dialog

There are two positions where information can be displayed for each branch (above and below.) In each of these positions, you can select to display information about the game tree (such as branch numbers or action names), or information about the currently selected solution (such as action probabilities or action values). Similarly, there are three positions at which information can be displayed for a node (above, below, and to the right). In the positions above and below the node, you can select to display information about the game tree (such as infoset name or ID), or information about the currently selected solution (such as realization probabilities, node values, belief probabilities). The position to the right of the node is reserved for information relating to
outcomes, and here you can select to display the outcome name or outcome vector.

**Table Window**

The table window is used several places in GAMBIT. It offers a large number of possible configuration options. Gambit will attempt to choose the most suitable settings for each use, but since Gambit runs on many different platforms, the default settings may not be suitable for your platform. The following settings can be set:

- **Font**: Selects the font to be used for the Labels and/or Data in the display.

- **Column Width**: The program will usually select the cell width to fit the widest item in the table. However, if an unusually long name is used, this width may be insufficient. The scrollbar controls the width of the cell, measured either in character width’s or pixels. Character based sizing is default and is recommended. If for some reason more precise dimensioning is desired, or if the cell width is not to change with the font size, the char checkbox should be unchecked. The width can be changed for a single column by choosing the desired column # in the Col choicebox or for all the columns by choosing All in the choicebox (default).

- **Show Labels**: It is possible to turn off the row and column labels by unchecking their corresponding checkboxes. However, this is not recommended since frequently important information is contained in those labels.

- **Color Text**: Some instances of the table window employ colored text for clearer presentation of the data (i.e. the normal form, solution inspection windows). The only reason to turn this feature off is to speed up display on very slow computer or over very slow networks.

- **Decimal Places**: The number of digits displayed after the decimal point in all floating point numbers is controlled by this slider. Note that this setting does not in any way effect the actual precision of the calculations.

All of these features can be saved to a defaults file (gambit.ini). They will be taken into account the next time the table window is used, as long as they are not overridden by the program.

**Mathematical Errors**

Under certain conditions some of the numerical algorithms may attempt to execute an invalid mathematical operation. Possible examples include taking a logarithm of 0, attempting to exponentiate a very large number, and other overflow or domain violation errors. When this condition is detected, the user is presented with an option to:

- **Continue**: This is the default option. The user acknowledges the error and the algorithm continues.

- **Ignore**: The user acknowledges the error and all future errors are ignored (no notification).
 Quit The user instructs the program to terminate.

Note that if either of the first two options is selected, the results returned by the algorithm may be invalid.

Incorrect Solutions

If you get a solution that looks incorrect, for example with probabilities that add to more than one, a possible cause is that the game has been solved in Rational precision, and the field width for the probabilities is not wide enough to display both the numerator and denominator of the corresponding rational number. This can happen, for example, when you solve in rational precision, but the data for the game is in floating point. Because of random digits in the insignificant final digits of a floating point number, the corresponding rational number may have very large numerators and denominators. Try any of the following to resolve this problem:

- Enter the data of the problem as rational precision (for example 19/10 instead of 1.9).

- Adjust the width of the columns for displaying the probabilities. This can be done using the Config button for either the Normal (p. 18) or Extensive (p. 19) Solution Windows. Unfortunately, it is very clumsy to do this in the current version, as it has to be done one column at a time. In the Normal form display, the column widths can be adjusted in the Prefs->Display->ColumnWidths menu item.

- Select the Float precision option in the Standard Solution dialog, and it will give solutions in floating point precision.

- Use the GCL instead of the GUI.

Another possible problem that can lead to apparently incorrect solutions is if the game is solved in floating point precision, but the current number of decimals displayed is set to zero. This can be adjusted in the Prefs->Display->Decimal Places menu item.

If you obtain what looks to be an incorrect solution, and it is not accounted for by either of the above, please report this to us, along with instructions on how to reproduce what you obtained. See the section on Technical Support (p. 9) for instructions on reporting bugs.
Normal Form GUI

The following is an example of the Normal form display of the sample poker game.

**Figure 22**: A 2 player normal form

**Figure 23**: NF GUI toolbar

**Normal Form Display**

In the normal form representation, the game is viewed as a 2-dimensional window into an N-dimensional matrix. Each 2D window shows a table of payoffs for each player as a function of the strategies for two of the players, holding the strategies of all of the other players (if there are any) fixed. Note that this organization requires each game to have at least two players. Each cell contains the payoff vector with one entry per player for the strategy profile determined by this cell. The profile itself is set by using a combination of settings for the row and column players plus the strategy settings for the rest of the players.
The payoff vector can be edited by double-clicking on the cell and entering new values in the dialog. To enter an entire game matrix, an accelerator key may be useful. Assuming the default accelerators are used, pressing the TAB key will move the cursor to the next cell of the normal form, and call up the payoff edit dialog. So an entire matrix can be entered by first double clicking on the first (upper left) cell, editing the entries (using tab to advance to successive entries), hitting return, then successively pressing tab, editing the entries and pressing return to enter the remaining cells of the matrix.

Row and Column Players

The row and column player choice boxes determine which players' strategies get displayed on the horizontal and vertical axis of the matrix. You can select a new player by clicking on the arrow at the right of the box, and selecting a new player from the list of players in the game. The dimensions of the matrix are determined by the number of strategies for the row and column players. Note that it is meaningless to select the same player for both the row and the column. If the game has more than two players, a warning will be issued and no action will be taken. In the case of a two player game, the row and the column players will be switched.

Strategy Profile

The array of choice boxes labeled "Profile" at all times reflects the strategies picked by each player to achieve the payoffs shown in the highlighted cell. The nth choice box can contain a value from 1 to the total number of strategies the nth player has. When one of the choice boxes is changed, one of two things can happen:

1. If the choice box number is not equal to either the row or the column player, the entire matrix will be updated with new values to reflect the new 2D view into the matrix.

2. If the choice box number was either the row or the column player, the highlighted cell will move to reflect the new strategy.

Normal Form Menu

File Menu (nfg)

Basic input/output operations are handled in this menu item.

Save: Saves the normal form game to a file. Normal form games are written in a standard format and use the extension .nfg. Note that the file format is forward, but not backward compatible: Previous versions of Gambit may not be able to read files written by this version, but this version should be able to read files written by previous versions of Gambit.

Output: Outputs the normal form game. The Output Media (p. 22) dialog is used to select where to send the output.
**Close:** Closes the normal form gui window, along with all child windows that depend on it.

**Edit Menu (nfg)**

This menu item is used to edit the normal form game.

**Label:** Change or enter a label for the normal form game. This label is saved along with the game if it is saved into a .nfg file, and appears in the window banner to identify the game that has been loaded.

**Players:** Edit the names assigned to the players.

**Strategies:** Assign labels to strategies for each player in the game.

**Outcomes:** Create, edit and delete outcomes

- **New:** Creates a new outcome for the game, and brings up the Change Payoff (p. 23) dialog to edit the payoffs for the outcome. Note that outcomes need not be attached to any cell. Use the attach item in this sub menu to attach the outcome to a cell in the game.

- **Delete:** Deletes the selected outcome. The Select Outcome (p. 24) dialog is used to select the outcome to delete.

- **Attach:** Brings up the Select Outcome (p. 24) dialog to select an outcome to attach to the highlighted cell of the normal form.

- **Detach:** Detaches the outcome from the highlighted cell of the normal form, resulting in no outcome attached to the cell.

- **Payoffs:** Edit the payoffs of the outcome attached to the highlighted cell of the normal form. The Change Payoff (p. 23) dialog is used to edit the payoffs.

**Supports Menu (nfg)**

Several menu items are available for manipulating Normal form supports.

- **Undominated:** Create new supports via elimination of dominated strategies. This brings up the dominance elimination (p. 27) dialog for selection of the type of dominance elimination to use.

- **New:** Used to create a new support for the normal form game. This brings up the Edit Support (p. 28) dialog to select the strategies for each player. For a detailed explanation of a Support see the section on supports (p. 57) in the theory section.
**Edit:** Edit the current support to add or remove strategies from it. The *Edit Support* (p. 28) dialog is used.

**Delete:** Delete a support from the list of supports.

**Select:** Select a support from the list as the current support.

**Solve Menu (nfg)**

This menu item provides access to the algorithms for solving the game.

**Standard:** Uses standard settings to compute various types of equilibria on the game. The *Standard Solution* (p. 23) dialog is used to select the type of equilibrium to compute.

**Custom:** Allows the user to select a particular algorithm and customize the settings for that algorithm. Only a user familiar with the algorithms used by Gambit should use the custom settings, since the program makes no attempt to check for the validity of some of the parameters. For a detailed discussion of the algorithms (p. 58) see the appropriate sections.

Not all algorithms are applicable to all games. If an algorithm is not available for the currently loaded normal form, it will be grayed out in the menu. When a custom algorithm is selected, a customized *Algorithm Parameters* (p. 38) dialog will be displayed to allow for selection of parameters for the algorithm.

- **EnumPure:** Finds all pure strategy equilibria of the game via *Enumeration* (p. 59).

- **EnumMixed:** Enumerates all mixed strategy equilibria of the game via the *EnumMixed* (p. 58) algorithm. Only valid for two person games.

- **Lcp:** Uses the Lemke-Howson, or *Linear Complementarity* (p. 61) method to find all accessible Nash equilibria. Only valid for two person games.

- **Lp:** Uses the *Linear programming* (p. 63) method. Only valid for two person zero sum games.

- **Liapunov:** *Liapunov* (p. 62) method.

- **Simpdiv:** *Simplicial subdivision* (p. 64).

- **Polenum:** *Polenum* (p. 63) method.

- **Qre:** Finds equilibria by Following a branch of the *Quantal Response Equilibrium* (p. 59) correspondence.
**QreGrid:** Computation of full QRE correspondence via a grid search (p. 60) (only feasible for very small games).

**View Menu (nfg)**

The normal form View menu item allows selection of what information to display on the normal form. The possible selection are

- **Solutions:** Selecting this menu item brings up the normal form Solutions Inspect (p. 18) dialog if there are any solutions in the current solution list.

- **Dominance:** A toggle to display dominance information about strategies in the normal form. When on, dominance information for each strategy is indicated along the border of the normal form. An $N$ means that the strategy is not dominated, $W$ means that it is weakly dominated, and $S$ means it is strongly dominated. Computations are with respect to the currently displayed support.

- **Probabilities:** A toggle to display the strategy probabilities for the currently selected solution. An extra row and column are added to the matrix to show the calculated probability of row and column players choosing the respective strategies. If there are more than two players, the number in the lower right cell is the probability of players other than row and column choosing their strategies to achieve this profile.

- **Value:** A toggle to display for each strategy the expected payoff for that strategy under the current solution.

- **Outcomes:** Toggle to select what to display in the cells of the normal form matrix. When on, the outcome name is displayed. When off, the payoff vector is displayed.

- **GameInfo:** Displays information about the game -- the number of players, whether or not it is constant sum, and indicates if the game has been edited since loading.

**Prefs Menu (nfg)**

The Prefs menu item is used to change various display features of the normal form matrix:

- **Display:** Used to change the number of decimal places that are displayed and the column width of the cells in the normal form.

- **Font:** Brings up a dialog to select the size and style of the font used to display data in the normal form.

- **Colors:** Used to select colors assigned to each player.
**Accels:** Accelerator keys can be defined to serve as shortcuts for frequently used commands. The *Accelerator Keys* (p. 26) dialog is used to define accelerator keys.

**Save:** Saves the current selections as the default. Unless saved, the selections in this menu item will return to the original values after the normal form is dismissed.

**Load:** Loads the current default selections for parameter selections set in this section.

**Normal Form Solutions**

A large number of different algorithms (p. 58) are available for solving normal form games. In order to cater to experienced users familiar with the properties of individual algorithms, many relevant parameters can be edited through a variety of solution related dialog boxes. To allow simpler access to the basic features provided by the algorithms, a set of commonly used standard solution types has been included with Gambit. The user need only determine the general features of the desired equilibria and Gambit will take care of all the details.

The standard solution settings are accessed via the *Solve->Standard* menu, or pressing the button on the toolbar.

If the current game was generated from an extensive form game, and if no changes have been made to either game, there will exist a link between the two forms. In this case, the *Extensive form* checkbox will be enabled. Checking this box will cause all generated solutions (if applicable to the solution type) to be converted back into behavior profiles and projected back to the extensive form window. The solutions can then be examined in the extensive form display.
NFG Standard Solutions

The standard solutions dialog is accessed via the Solve->Standard menu. The particular settings are then picked in the dialog box. You can select what kind of an equilibrium to compute (Nash or Perfect), and a maximum number of solutions to compute. When you select a standard solution setting, it automatically sets various parameters that control the solution computation. These parameters include whether and how to iteratively eliminate dominated strategies, which algorithm to use for the computation, and the maximum number of solutions to search for before finishing the computation. The settings for each of the standard solutions are given in the following table:

<table>
<thead>
<tr>
<th>Type</th>
<th>Number</th>
<th>Game Type</th>
<th>UnDominated</th>
<th>Algorithm</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nash</td>
<td>One</td>
<td>2 person constant sum</td>
<td>Weak</td>
<td>LP-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 person general sum</td>
<td>Weak</td>
<td>LCP-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n person</td>
<td>Weak</td>
<td>SimpDiv-1</td>
<td></td>
</tr>
<tr>
<td>Nash</td>
<td>Two</td>
<td>2 person n person</td>
<td>Strong</td>
<td>EnumMixed-2</td>
<td>Liap-2</td>
</tr>
<tr>
<td>Nash</td>
<td>All</td>
<td>2 person n person</td>
<td>Strong</td>
<td>EnumMixed-0</td>
<td>PolEnum-0</td>
</tr>
<tr>
<td>Perfect</td>
<td>One</td>
<td>2 person n person</td>
<td>Weak</td>
<td>LCP-1</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Perfect</td>
<td>Two</td>
<td>2 person n person</td>
<td>Weak</td>
<td>EnumMixed-2</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Perfect</td>
<td>All</td>
<td>2 person n person</td>
<td>Weak</td>
<td>EnumMixed-0</td>
<td>Not implemented</td>
</tr>
</tbody>
</table>

In the entry for the algorithm for the above table, after the algorithm name is indicated the maximum number of solutions to be found. If this number is 0, the algorithm will only terminate when it has found all solutions. For algorithms which are not guaranteed to find all solutions (such as Liap), the algorithm will enter an infinite loop, and continue to search until the computation is aborted by the user. To have individual control over each any of the parameters mentioned above, see the Custom Solutions (p. 38) section.

NFG Custom Solutions

Only a user familiar with the algorithms should use the custom settings since the program makes no attempt to check for the validity of some of the parameters. For a detailed discussion of the algorithms (p. 58) see the appropriate sections.

A custom algorithm is selected in the Solve->Custom submenu. Each solution algorithm is controlled by two groups of settings, both of which are set on the dialog box for the particular algorithm.

**Dominance elimination**: Prior to running the algorithm, dominated strategies may be eliminated, thus speeding up the consequent calculations. The type of
dominance elimination is controlled by three radioboxes at the top the dialog box for the algorithm.

**Depth:** Determines whether and how much dominance elimination is to be done. **None** indicates no dominance elimination, **Once** selects a single step of elimination, and **Iterative** selects iterative elimination of dominated strategies.

**Type:** Determines whether to use **weak** or **strong** domination.

**Method:** Determines whether to eliminate strategies that are dominated only by pure strategies or whether to also eliminate strategies dominated by mixed strategies. Elimination by pure strategies is faster, but may not find all dominated strategies, as it is possible that strategies that are not dominated by any pure strategy may be dominated by a mixed strategy.

**Algorithm parameters:** The remainder of the dialog box is used for setting parameters that are specific to the algorithm. These parameters will differ from algorithm to algorithm. The settings for the particular algorithm are described in the *algorithms* (p. 58) section.

Note that some algorithms require a starting point. Depending on the options selected in the custom parameters dialog for these algorithms, a starting point dialog may pop up before the algorithm is run. The default (centroid) is usually the best place to start. Double clicking on the first profile in the window and pressing **Ok** will select the default and allow the algorithm to continue. Advanced users may choose to enter a different starting point, such as profiles resulting from the use of other algorithms.

**Default Accelerator Keys**

To speed up some of the more commonly used commands in the GUI, a set of **accelerator keys** may be defined. Once defined, a combination of keys can be used to activate most of the menus and buttons. Accelerator key definitions are accessed through the Prefs->Accels menu. Accelerator key definitions are saved in the main Gambit defaults file (gambit.ini). Gambit is distributed with the following keys defined.

**TAB:** The TAB key is defined to be **EditNextPayoff** event, and can be used to rapidly enter or modify a normal form. Pressing TAB will automatically move the cursor to the next cell in the normal form, and display a dialog box to edit the entries. Thus the entire game can be entered by repeatedly pressing TAB and entering the values -- no mouse operations are required.

**RETURN:** The RETURN key is defined as teh **EditPayoff** event. Hitting **RETURN** will bring up a dialog box to enter or edit the payoff of the outcome attached to the selected cell of the normal form.
Extensive Form GUI

In the extensive form, the game is represented as a topological tree. The section on representation of the Extensive Form (p. 11) has a more detailed explanation of this form. Compared to the Normal Form, the Extensive Form interface is much richer and thus considerably more complex. A major portion of the functionality is devoted to the tree building. Another set of functions deals with customizing the display, and yet another set of functions takes care of the solutions and their display.

Extensive Form Display

Navigating the Extensive Form

The cursor is indicated on the display of the normal form by a solid or flashing dark line above one of the nodes or by a dark triangle around a subgame icon. The cursor position is used by many of the tree building functions. You can move the cursor around the extensive form game either by use of the arrow keys or by using the mouse.
When the extensive form game becomes large, only a window into the extensive form game is displayed. As you move the cursor around the game tree, the window will redraw the game tree when necessary to keep the cursor visible.

For large games, the whole game tree will not be visible at one time. If you want to see more of the game tree, you can change the magnification by using the Prefs - > Zoom menu. You can also use the – and + keys to change the magnification by constant increments. When editing a large game, if the magnification level is too small, you may not be able to read the textual information displayed on the game tree. You can use the Zoom Window, which can be enabled in the Inspect - > ZoomWindow menu to get a magnified view of the tree at the cursor location.

Drag and Drop

Many of the editing functions can be executed by drag-and-dropping with the mouse.

Figure 26: Sample Drag n' Drop operations

Most of the labeling information can be quickly changed by double-clicking at the location on the tree of the label and entering the data in the dialog that pops up. Similarly, double clicking on the location of an outcome will bring up the payoff dialog for outcomes.

Extensive Form Menu
File Menu (efg)

Basic input/output operations are handled in this menu item.

**Save:** Saves the extensive form game to a file. Extensive form games are written in a standard format and use the extension `.efg`. Note that the file format is forward, but not backward compatible: Previous versions of Gambit may not be able to read files written by this version, but this version should be able to read files written by previous versions of Gambit.

**Output:** Outputs the extensive form game. The *Output Media* (p. 22) dialog is used to select where to send the output.

**Close:** Closes the extensive form gui window, along with all child windows that depend on it.

Edit Menu (efg)

This menu item is used to build or modify the extensive form game. This section assumes a working knowledge of the conventions used in the GAMBIT extensive form display (discussed in the *Extensive Form* (p. 11) section). Note that any structural changes to the tree structure will invalidate all existing solutions. The solutions list will be cleared and solutions removed.

The *Edit* menu may also be called up by clicking the right mouse button causing the menu to pop up at the current mouse cursor position.

![Figure 27: Popup Edit menu](image)

In addition, many of the operations in this menu can be implemented by *drag and drop* (p. 41) operations, as indicated in the descriptions below.

**Node:**

**Add Move:** The selected node is put into an information set using the *Add Move* (p. 28) dialog. The result is that the node becomes a decision node, with number of branches equal to the number of actions in the
information set. New child nodes are created as successor nodes to the actions in the information set. Only valid at terminal nodes.

**Delete Move:** Deletes the move at the cursor node from the extensive form game. The subtree rooted at one of the actions takes the place of the deleted node in the tree. You will be prompted to select a branch to keep. All other subtrees descending from the cursor node are deleted. Deleting a move does not delete the information set which the corresponding node was in.

**Insert Move:** This is analogous to Add Move (p. 28), except that it can be used at a non terminal node. The move is inserted into the tree at that point, and the portion of the tree that was previously at that node is moved to the first child of the inserted node.

**Label:** Each node can have a label associated with it. This function allows the entering or modification of this label. The display of these labels is controlled in the **Legends** (p. 29) Dialog.

**Set Mark:** The GUI allows you to mark (memorize) one node for later use. The marked node is required in some tree operations. The marked node is indicated by a small circular token on the game tree. Note that drag-and-drop mouse operations make the node-mark unnecessary.

**Goto Mark:** Moves the cursor to the mark. This can be useful in a very large game to quickly move from one part of the game to another.

**Actions:**

**Delete:** Select the action to be deleted from the dialog box. The selected action will be removed from all information sets that it belongs to. All descendants of the nodes at this branches are also removed.

*Drag-And-Drop:* (p. 41) Place the cursor on top of a branch corresponding to the action to be deleted while pressing the shift key. The cursor will change to a scissors token. Pressing the mouse button while the scissors are displayed will delete the action from the tree.

**Insert:** A new action is inserted after the action that is selected from the dialog box. A branch will be added to every node that belongs to the same information set as the selected node.

*Drag-And-Drop:* (p. 41) Place the cursor at the juncture of the branch lines with the end of the node line (the point where the action branches begin). Press the mouse button and drag it to the right. A new branch will appear as a "rubber band," which you can drag to a location where you want the new action to appear.

**Append:** Appends a new action as the last action at the selected information set.
Label: Each action in an infoset can have a label. This allows the setting or changing of this label.

Probabilities: Allows the explicit setting of each action probability. Only valid when the cursor is on a CHANCE node.

InfoSet:

Merge: Merges the information set at the cursor into the information set at the mark. Both information sets must have the same number of actions. Note that you cannot merge infosets across different subgames (unmark subgames first).

Drag-And-Drop: (p. 41) The merge operation can be achieved by clicking on an infoset marker (small circle) at the base of a node and dragging it to the infoset marker of the target node.

Break: Removes the node at the cursor from its information set, creating a new information set with just that node in it.

Split: Splits the information set with all of the nodes above the cursor node remaining in the same infoset, while all of the nodes below the cursor node are moved to a newly created infoset.

Drag-And-Drop: (p. 41) The split operation can be achieved by moving the mouse cursor on top of the line connecting the nodes in the infoset while holding down the Shift key. When the mouse cursor is on top of the infoset line, it will change to 'scissors.' Pressing the mouse button while the cursor is 'scissors' will split the infoset.

Join: Removes the node at the cursor from its information set, and puts it into the information set at the mark.

Label: Assigns a textual name to the information set. These names can be displayed on the extensive form display by appropriate selection in the (p. Error! No bookmark name given.) section.

Player: This allows changing the player that has the choice at this node. You can choose any player except the one that is currently selected. If player names are being displayed on the tree, double clicking on the name will also activate this operation.

Reveal: Reveals the information set to selected players. This will result in refining the information partitions of players to which the information set is revealed in such a way that they observe all actions at the information set.
**Outcomes:** Create, edit and delete outcomes

**New:** Creates a new outcome for the game, and brings up the *Change Payoff* (p. 23) dialog to edit the payoffs for the outcome. Note that outcomes need not be attached to any node. Use the attach item in this sub menu to attach the outcome to a cell in the game.

**Delete:** Deletes the selected outcome. The outcome will be also detached from any nodes in the extensive form game to which it was attached. The *Select Outcome* (p. 24) dialog is used to select the outcome to delete.

**Attach:** Brings up the *Select Outcome* (p. 24) dialog to select an outcome to attach to the selected node of the extensive form.

**Detach:** Detaches the outcome from the selected node, resulting in no outcome attached to the node.

**Label:** Assign a label or name to the outcome attached to selected node.

**Payoffs:** Edit the payoffs of the outcome attached to the selected node of the extensive form. The *Change Payoff* (p. 23) dialog is used to edit the payoffs. If there is no outcome attached, then a new outcome is created and attached to the node.

**Tree:**

**Copy** Copies the part of the tree following the mark to the position of the cursor. Note that the cursor must be at a terminal node. No new information sets are created by this operation. Thus, you may have to edit the information sets after this operation to achieve the desired change.

*Drag-And-Drop:* (p. 41) The copy function can be achieved by clicking on and dragging the source node to the target node while holding down the Control key. The cursor will change to a small 'tree' icon with the word 'copy' to indicate that a copy operation is in progress. Releasing the mouse while the cursor is not on any terminal node will cancel the operation.

**Move** Moves the part of the game tree following the mark to the position of the cursor.

*Drag-And-Drop:* (p. 41) The move function can be achieved by clicking on and dragging the source node to the target node. The cursor will change to a small 'tree' icon with the word 'move' to indicate that a move operation is in progress. Releasing the mouse while the cursor is not on any terminal node will cancel the operation.
**Delete**  The node at the cursor and all its descendants will be deleted.

*Drag-And-Drop: (p. 41)* The Delete Tree function can be achieved by placing the cursor on the node line for the sub tree that you want to delete, while pressing the shift key. The cursor will change to a scissors token. Pressing the mouse button while the scissors are displayed will delete the node and all its descendants from the tree.

**Label**  The label pertains to the game as a whole and will be displayed on the titlebar of the window.

**Players**  Each player must have a name. Although not required, it is highly recommended that these names be unique. When a player is first created, it is given a default name "Player #." This dialog allows you to change these default names to something appropriate to the game.

**Infoset**  This dialog allows you to change the names of information sets, remove empty information sets, and create new information sets.

### Subgames Menu (efg)

The following list describes the commands for working with subgames.

**Mark All:** Marks all subgames. Note that the subgame roots now become marked by small rectangles.

**Mark:** Check whether the selected node is a subgame root, and mark it if it is. If the selected node is not a root of a valid subgame, a warning is issued.

**UnMark All:** Unmarks all subgames, consequently expanding all subgames.

**UnMark:** If the cursor node is a marked subgame, that subgame is removed from the list of subgames and is no longer considered either for display or solution purposes. If the cursor node is not a subgame root, a warning is issued.

**Collapse Level:** If the cursor node is a subgame root, that subgame is collapsed and all of the nodes contained in it are replaced with a subgame icon. The cursor changes shape to reflect this change.

**Collapse All:** Applies Collapse to every node that is marked as a subgame root, including the ROOT node.

**Expand Level:** If the cursor node is a subgame root, that subgame is expanded and all of the nodes, up to the first descendant collapsed subgame, are displayed.

**Expand Branch:** If the cursor node is a subgame root, all of the descendant subgames are expanded.

**Expand All:** Applies Expand to every node that is marked as a subgame root.
Supports Menu (efg)

Several menu items are available for manipulating Normal form supports.

**Undominated:** Create new supports via elimination of dominated strategies. This brings up the *dominance elimination* (p. 27) dialog for selection of the type of dominance elimination to use.

**New:** Used to create a new support for the extensive form game. This brings up the *Edit Support* (p. 28) dialog to select the strategies for each player. For a detailed explanation of a *Support* see the section on *supports* (p. 57) in the theory section.

**Edit:** Edit the current support to add or remove strategies from it. The *Edit Support* (p. 28) dialog is used.

**Delete:** Delete a support from the list of supports.

**Select:** Select a support from the list as the current support.

**Root Reachable:** A toggle which determines how to draw an extensive form without full support. If selected, only the nodes that are reachable from the root node are drawn. If not selected, then all nodes are drawn, but only the branches corresponding to actions in the support are drawn.

Solve Menu (efg)

**Standard:** Uses standard settings to compute various types of equilibria on the game. The *Standard Solution* (p. 23) dialog is used to select the type of equilibrium to compute.

**Custom:** Allows the user to select a particular algorithm and customize the settings for that algorithm. Only a user familiar with the algorithms used by Gambit should use the custom settings, since the program makes no attempt to check for the validity of some of the parameters. For a detailed discussion of the *algorithms* (p. 58) see the appropriate sections.

Not all algorithms are applicable to all games. If an algorithm is not available for the currently loaded normal form, it will be grayed out in the menu. When a custom algorithm is selected, a customised *Algorithm Parameters* (p. 53) dialog will be displayed to allow for select parameters for the algorithm.

**Extensive Form:** Algorithms that do computations directly on the extensive form of the game.

**EnumPure:** Finds all pure strategy equilibria of the game via
Enumeration (p. 59).

**Lcp:** Uses the Lemke-Howson, or *Linear Complementarity* (p. 61) method to find all accessible Nash equilibria. Only valid for two person games.

**Lp:** Uses the *Linear programming* (p. 63) method. Only valid for two person zero sum games.

**Liapunov:** *Liapunov* (p. 62) method.

**Polenum:** *Polenum* (p. 63) method.

**Qre:** Finds equilibria by Following a branch of the *Quantal Response Equilibrium* (p. 59) correspondence.

**Normal Form:** Algorithms that do computations by converting the game to normal form.

**EnumPure:** Finds all pure strategy equilibria of the game via *Enumeration* (p. 59).

**EnumMixed:** Enumerates all mixed strategy equilibria of the game via the *EnumMixed* (p. 58) algorithm. Only valid for two person games.

**Lcp:** Uses the Lemke-Howson, or *Linear Complementarity* (p. 61) method to find all accessible Nash equilibria. Only valid for two person games.

**Lp:** Uses the *Linear programming* (p. 63) method. Only valid for two person zero sum games.

**Liapunov:** *Liapunov* (p. 62) method.

**Simpdiv:** *Simplicial subdivision* (p. 64).

**Polenum:** *Polenum* (p. 63) method.

**Qre:** Finds equilibria by Following a branch of the *Quantal Response Equilibrium* (p. 59) correspondence.

**QreGrid:** Computation of full QRE correspondence via a *grid search* (p. 60) (only feasible for very small games).

**Normal Form:**

**Reduced:** Constructs the reduced normal form of the game, and displays it in the Normal Form GUI.

**Agent:** Constructs the agent normal form of the game, and displays it in
the Normal Form GUI.

Inspect Menu (efg)

This menu item allows the selection of information to display:

**Solutions:** Displays the extensive form Solutions Inspect (p. 19) dialog if there are any solutions in the current solution list.

**Cursor:** Displays the Inspect Node (p. 25) dialog which shows information about solution values for the selected node. The displayed information continually updates as the extensive form is navigated.

**Infosets:** When enabled, double clicking on an infoset in the EF window will highlight the corresponding infoset in the Solution inspect window. Double clicking on an infoset (Iset column) in the Solution inspection window will highlight the corresponding infoset in the EF window.

**Zoom Window:** Brings up the Zoom Window (p. 26), which provides a window into the game, centered around the selected node.

**Game Info:** Displays selected information about the game, such as the number of players, whether the game is Perfect Recall and whether it is Constant Sum.

Prefs Menu (efg)

The prefs menu allows you to load, set and save various parameters which affect the extensive form display.

**Display:** This menu you to change parameters affecting the way in which the game tree is drawn, such as the length of the nodes, branches, and vertical spacing between nodes, and also allows for setting whether to display information sets (by connecting the nodes) and whether to have a flashing cursor.

**Decimal Places:** Number of decimal places to display for the data in the extensive form.

**Flashing Cursor:** A toggle to control whether the cursor flashes.

**Layout:** Brings up a dialog to customize the various measurements in the rendering of the game tree.

**Legend:** Allows you to specify what information will be displayed on the extensive form display next to each node and next to each branch by selecting from the Legends (p. 29) dialog. *Note that double clicking on any of the textual information selected above will call up a dialog to modify the particular entry*

**Fonts:** Allows selection of the font size of information displayed on the game tree.
**Colors:** You can change the default colors assigned to the players in this menu.

**Accels:** You can define and save accelerator keys for use in the extensive form. A more complete discussion of this topic is given in the section on *Accelerator Keys* (p. 26)

**Save:** Saves the current settings of the parameters in this menu. These settings will become the default values.

**Load:** Loads the default settings of the parameters for this menu.

**Extensive Form Solutions**

A large number of different algorithms (p. 58) are available for solving extensive form games. In order to cater to experienced users familiar with the properties of individual algorithms, many parameters can be edited through a variety of solution related dialog boxes. To allow simpler access to the basic features provided by the algorithms, a set of commonly used standard solution types has been included with Gambit. The user need only determine the general features of the desired equilibria and Gambit will take care of all the details.

All of the settings are saved to a defaults file every time they are changed. There is no need to enter the custom parameters before each recalculation, even through multiple sessions, unless a change is required. After selecting the solution type/parameters, the algorithm must be started by selecting the Solve->Solve menu or pressing the button on the toolbar.

**Subgames**

Gambit implements and supports the concept of a game theoretical subgame. For the conditions necessary for a subtree to be qualified as a subgame, see [4]. Gambit can be made to selectively recognize some, none, or all of the legal subgames in its solution algorithms through the technique of marking subgames.

When an extensive form is first created or read in from a file, no subgames (except for the default ROOT subgame) are marked. If a node is the root of a marked subgame, a small triangle is drawn at its base. Subgames can be collapsed for the purpose of better viewing a large tree. If a subgame is collapsed, all of the nodes it contains are not displayed but replaced by a subgame icon. Shift-clicking (holding down the *SHIFT* key while clicking the left mouse button) on a subgame root node will toggle the state of that subgame (expanded vs. collapsed). Refer to the following picture for an example of subgame structures.
All of the solution algorithms (with the current exception of the QRE algorithms) make use of the marked subgames. When non-trivial subgames are defined, then the solution algorithms will solve the extensive form game by recursion through the marked subgames. Thus, a subgame is solved only when all subgames following it have been solved.

If all subgames are marked, then any Nash equilibrium found will be a subgame perfect Nash equilibrium. Note that the solution algorithms only respect marked subgames, and hence if you want the solution algorithms to make use of subgames, you must mark the subgames. When an algorithm encounters a subgame, the subgame is solved by itself and the rest of the tree is resolved for each of the subgame solutions. Clearly, the total number of solutions grows exponentially with the number of subgames. For advanced users, the Solve->Custom->Extensive->Algorithm dialog provides the option to interactively select which solutions are used for each subgame. This feature lets the advanced user concentrate on solutions of interest while keeping the total number of solutions small.

**Extensive form Supports**

The concept of an extensive form support is very similar to that of a normal form support. Instead of considering the entire game tree, a subset of all the player actions is considered. That is, at each decision node the player may have a reduced number of choices from that of the original tree (full support). However, at least one action must
exist at each node. Most of the extensive form solution algorithms support, the notion of extensive form supports.

An extensive form support can be generated in one of two ways--either by dominated strategy elimination or by explicit creation of a new support. To eliminate dominated strategies in an extensive form game, you use the Support->UnDominated (p. 27) menu item.

To create a support, use the Edit Support (p. 28) dialog, which is accessible via the Supports->New menu item.

**EFG Standard Solutions**

The standard solutions are selected by the Solve->Standard menu. The particular settings are then picked in the dialog box. You can select a refinement of Nash equilibrium to compute (Nash, Subgame perfect, and Sequential), and a maximum number of solutions to compute. When you select a standard solution setting, it automatically sets various parameters that control the solution computation. These factors include whether to mark subgames before solving, whether and how to iteratively eliminate dominated strategies, which algorithm to use for the computation, and the maximum number of solutions to search for before finishing the computation. The settings for each of the standard solutions are given in the following table:
<table>
<thead>
<tr>
<th>Type</th>
<th>Number</th>
<th>Game Type</th>
<th>Subgames</th>
<th>UnDominated</th>
<th>Algorithm</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nash</td>
<td>One</td>
<td>2-person</td>
<td>Mark</td>
<td>Weak</td>
<td>EF/LP-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>constant sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-person</td>
<td>Mark</td>
<td>Weak</td>
<td>EF/LCP-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>general sum</td>
<td>Mark</td>
<td>Weak</td>
<td>NF/SimpDiv-1</td>
<td></td>
</tr>
<tr>
<td>Nash</td>
<td>Two</td>
<td>2-person</td>
<td>Unmark</td>
<td>Strong</td>
<td>NF/EnumMixed-d-2</td>
<td>Not guaranteed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n-person</td>
<td>Unmark</td>
<td>Strong</td>
<td>EF/Liap-2</td>
<td></td>
</tr>
<tr>
<td>Nash</td>
<td>All</td>
<td>2-person</td>
<td>Unmark</td>
<td>Strong</td>
<td>NF/EnumMixed-d-0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n-person</td>
<td>Unmark</td>
<td>Strong</td>
<td>NF/PolEnum</td>
<td></td>
</tr>
<tr>
<td>Subgame Perfect</td>
<td>One</td>
<td>2-person</td>
<td>Mark</td>
<td>Weak</td>
<td>EF/LP-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>constant sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-person</td>
<td>Mark</td>
<td>Weak</td>
<td>EF/LCP-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>general sum</td>
<td>Mark</td>
<td>Weak</td>
<td>NF/SimpDiv-1</td>
<td></td>
</tr>
<tr>
<td>Subgame Perfect</td>
<td>Two</td>
<td>2-person</td>
<td>Mark</td>
<td>Strong</td>
<td>NF/EnumMixed-d-2</td>
<td>Not guaranteed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n-person</td>
<td>Mark</td>
<td>Strong</td>
<td>NF/Liap-2</td>
<td></td>
</tr>
<tr>
<td>Subgame Perfect</td>
<td>All</td>
<td>2-person</td>
<td>Mark</td>
<td>Strong</td>
<td>NF/EnumMixed-d-0</td>
<td>Not guaranteed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n-person</td>
<td>Mark</td>
<td>Strong</td>
<td>EF/Liap-0</td>
<td></td>
</tr>
<tr>
<td>Sequential</td>
<td>One</td>
<td>2 or n-person</td>
<td>Unmark</td>
<td>None</td>
<td>EF/QRE-1</td>
<td></td>
</tr>
<tr>
<td>Sequential</td>
<td>Two</td>
<td>2 or n-person</td>
<td>Unmark</td>
<td>None</td>
<td>EF/Liap-2</td>
<td>Not guaranteed</td>
</tr>
<tr>
<td>Sequential</td>
<td>All</td>
<td>2 or n-person</td>
<td>Unmark</td>
<td>None</td>
<td>EF/Liap-0</td>
<td>Not guaranteed</td>
</tr>
</tbody>
</table>

In the entry for the algorithm for the above table, first is indicated whether the algorithm is applied to the extensive (EF) or normal (NF) form. Then the name of the algorithm is given. Finally, the number after the algorithm indicates the maximum number of solutions to be found. If this number is 0, the algorithm will only terminate when it has found all solutions. For algorithms which are not guaranteed to find all solutions (such as Liap), the algorithm will enter an infinite loop, and continue to search until the computation is aborted by the user.

**EFG Custom Solutions**

Only a user familiar with the algorithms should use the custom settings since the program makes no attempt to check for the validity of some of the parameters. For a detailed discussion of the algorithms (p. 58) see the appropriate sections.

A custom algorithm is selected in the **Solve->Custom** submenu. The game can be
solved either by algorithms that work directly on the extensive form, or by conversion to the normal form, and solving with algorithms on the normal form. In the later case, a normal form representation will be automatically created, the algorithm run and the solutions converted back to behavioral strategy profiles.

The sub menus Solve->Custom->Extensive and Solve->Custom->Normal are used to access algorithms which work on the extensive and normal forms respectively. Not all algorithms are applicable to all games (for example, Linear Programming and Linear Complementarity algorithms can only be applied to two person games.) Algorithms that are not applicable to the current game will be greyed out in the menu.

After selecting an algorithm, the Algorithm parameter dialog will appear. Each solution is controlled by three groups of settings, which appear on the dialog box for the algorithm in the following order:

**Dominance elimination:** Prior to running the algorithm, dominated strategies may be eliminated, thus speeding up the consequent calculations. For normal form algorithms, the domination is done on the reduced normal form after converting the game to normal form. For extensive form algorithms, dominance elimination is done directly on the extensive form (Note: dominance elimination on the extensive form can be slower than on the reduced normal form since it grows in complexity with the size of the agent normal form of the game.)

The type of dominance elimination is controlled by radioboxes at the top of the dialog box for the algorithm. In the case of algorithms that work on the normal form, the same options are available as is described in the normal form algorithm section:

**Depth:** Determines whether and how much dominance elimination is to be done. None indicates no dominance elimination is to be done, Once selects a single step of elimination, and Iterative selects iterative elimination of dominated strategies.

**Type:** Determines whether to use weak or strong domination.

**Method:** Determines whether to eliminate strategies that are dominated only by pure strategies or whether to also eliminate strategies dominated by mixed strategies. Elimination by pure strategies is faster, but may not find all dominated strategies, as it is possible that strategies that are not dominated by any pure strategy may be dominated by a mixed strategy. For algorithms that work directly on the extensive form, the Pure/Mixed selection is not available.

**Subgames:** Subgames (p. 46) may be marked prior to running any algorithm. If subgames are marked, Gambit will apply the selected algorithm to each subgame in reverse tree traversal order. All solutions at each subgame are used to generate continuation values for the higher level subgames. Marking subgames before running an algorithm that computes Nash equilibria will insure that all equilibria computed are subgame perfect Nash equilibria.
If there are a lot of subgames, and each subgame produces multiple solutions, the total number of solutions can grow exponentially. To interactively select which solutions to the subgames to keep, toggle the interactively select subgames checkbox.

**Algorithm parameters**  The remainder of the dialog box is devoted to parameters specific to the algorithm. Refer to the algorithm section for these parameters.

Note that some algorithms require a starting point. Depending on the options selected in the custom parameters dialog for these algorithms, a starting point dialog may pop up before the algorithm is run. The default (centroid) is usually a good place to start. Double clicking on the first profile in the window and pressing OK will select the default and allow the algorithm to continue. Advanced users may choose to use values obtained by solving the game using a different algorithm as a starting point.

**EFG Solutions and Subgames**

If the extensive form contains more than one subgame, and if the subgames are either marked explicitly or implicitly by selecting the Mark Subgames, Gambit will apply the selected algorithm to each subgame in reverse tree traversal order. For each solution obtained at a subgame node, a full solution will exist. If there are a lot of subgames, and each subgame produces multiple solutions, the total number of solutions grows exponentially.

To provide the user with interactive control over which subgame solutions will be used to compute the full solutions, the subgame solution picking dialog can be used. This option is enabled from the main Custom solutions dialog by checking the interactively select subgame solutions checkbox. When enabled, for each subgame solved by Gambit, a solution display window will be created and the user must select the subgame solutions to be used for further calculations. Solutions are selected/deselected by double-clicking or Ctrl-clicking on the corresponding solution number. At least one solution must be selected for each subgame.
Figure 29: Subgame solution picking dialog w/ all solns selected

To speed up solution selection two shortcuts exist:

**Select/unselect all of the solutions at once.** The dialog starts up with all of the solutions in the same state (selected or unselected) depending on the current default (see next item). If all of the solutions are selected, a None button will exist. Pressing the button will unselect all of the solutions. The button will then change to All. Pressing the All button will select all of the solutions.

**Default selection parameter.** The startup state of the solution picking dialog can be controlled through the Opt button. The Pick All Solutions item is saved to a defaults file and is used for all future dialogs.

**Default Accelerator Keys**

To speed up some of the more commonly used commands in the GUI, a set of accelerator keys may be defined. Once defined, a combination of keys can be used to activate most of the menus and buttons. Accelerator key definitions are accessed through the Prefs->Accels menu. Accelerator key definitions are saved in the main Gambit defaults file (gambit.ini). Gambit is distributed with the following keys defined.

**ENTER:** Marks the node (Edit->Node->Mark)

**SPACE:** Redraws the screen.
Solutions of Games

Supports

A support is a subset of the strategies or actions of a game which is of interest for some reason. Usually the reason for looking at a support that is less than the full support is to eliminate dominated strategies. All of the solution algorithms in Gambit take the current support as an argument. When starting analysis of any game, the current support is the full support of the game. New supports are created either through elimination of dominated strategies or explicit creation of a support. This functionality is accessed via the Support menu item in either the Normal Form (p. 34) or Extensive Form (p. 47) GUI.

Equilibria

Nash Equilibrium

Every finite game has at least one Nash equilibrium in either pure or mixed strategies ([5]).

Gambit has a number of algorithms to find Nash equilibria. The appropriate algorithm to use depends on a number of factors, most importantly, the number of players in the game, and the number of equilibria you want to find. For a detailed discussion of these issues, see [6].

Before computing equilibria, you may wish to eliminate dominated strategies (p. 27) from the support. The smaller the support, the faster any algorithm will run. However, dominance elimination can itself be computationally intensive, especially if mixed domination is used, or if dominance elimination is done on the extensive form.

- If you want to find more than one, or all Nash equilibria of a game, then you may first successively eliminate strongly dominated strategies. Any equilibrium of the original game will also be an equilibrium of the reduced game.

- If you just want to find one Nash equilibrium, you can first successively eliminate weakly dominated strategies. Elimination of weakly dominated strategies may eliminate some Nash equilibria of the original game (so it should not be used if you want to find multiple Nash equilibria,) but any Nash equilibrium to the reduced game will be an equilibrium to the original game, so it can be used if you only want to find one equilibrium.

Pure Equilibria

The EnumPure (p. 59) algorithm can be used to enumerate all of the pure strategy equilibria for both extensive and normal form games. select EnumPure, and check Use NF. This will find all pure strategy Nash equilibria of the associated reduced normal form of the game, and convert them to behavior strategies.
Two Person Constant Sum Games

For two person constant sum normal form games, the minimax theorem applies. The set of Nash equilibria is a convex set, and the problem of finding Nash equilibria can be formulated as a linear program. The $L_p$ algorithm (p. 63) will solve a constant sum game using this approach.

Two Person Games

For two person nonzero sum games, the problem of finding Nash equilibria can be formulated as a linear complementarity problem, and exact solutions can be found as long as computations are done in rationals. The $Lcp$ algorithm (p. 61) solves a two person game using this approach. Note that the $Lcp$ algorithm can also be used directly on an extensive form game, where it implements the Koller, Megiddo, von Stengel Sequence form [7]. To find one Nash equilibrium use $Lcp$ with nEquilib set to 1.

For a two person game the $EnumMixed$ algorithm (p. 58) will enumerate all of the extreme points of the components of the set of Nash equilibria, and hence can be used to find all Nash equilibria. Using $EnumMixed$ with nEquilib set to 0 will find all Nash equilibria. To find if there is more than one Nash equilibrium use $EnumMixed$ with nEquilib set to 2.

N Person Games

For n-person normal form games, with n greater than two, the $PolEnum$ (p. 63) algorithm will find all Nash equilibria. The $PolEnum$ algorithm may be computationally infeasible on large games. Thus other algorithms are also available for finding one or a sample of Nash equilibria. Since Nash equilibria can be irrational, the algorithms to locate one equilibrium will only find approximations (to machine accuracy) of Nash equilibria.

The $SimpDiv$ algorithm (p. 64) is guaranteed to locate one equilibrium for an n-person normal form game. This algorithm can be very slow on some games. Hence two other algorithms are also supported, $QRE$ (p. 59) and $Liap$ (p. 62). These algorithms can also be used to search for multiple equilibria (with no guarantee that all have been found,) and to search for equilibrium close to a given starting point.

sequential equilibrium

Sequential equilibria are equilibria that prescribe optimal behavior at any information set of the extensive form, given a consistent assessment of beliefs. See [8]. To compute an approximation to a sequential equilibrium you can select QRESolve in the Extensive form Solve menu.

Solution Algorithms

$EnumMixed$
Finds all Nash equilibria for a two person game. More precisely, it finds the set of extreme points of the components of the set of Nash equilibria. The procedure is to enumerate the set of complementary basic feasible solutions (see eg. /citeMan:64.)

Limitations: Only works for two-person normal form games.

The following parameters can be set:

- **Stop After**: Specifies the maximum number of equilibria to find. The default is zero, which means that all equilibria are found. To check if there is a unique Nash equilibrium, one could set this parameter to 2.

- **precision**: Select **Rational** for exact computations, **Float** for speed. See the section on Data Types (p. 17) for more information.

**EnumPure**

Computation of pure strategy Nash equilibria is done by simple enumeration. All pure strategies are checked to see if they are Nash equilibria.

The following parameters can be set.

- **Stop After**: Allows the user to set the maximum number of Nash equilibria to find.

**QRE**

Computes a branch of the logistic quantal response equilibrium correspondence for n-person normal form games (as described in [9] and n-person extensive form games (as described in [10]).) The branch is computed for values of $\lambda$ between minLam and maxLam. The algorithm starts at $\lambda(0) = \text{minLam}$ if delLam > 0, or $\lambda(0) = \text{maxLam}$ if delLam < 0. It then increments according to the formula

$$
\lambda(t+1) = \lambda(t) + \text{delLam} \cdot \lambda(t)^a.
$$

where minLam, maxLam, delLam, and a are parameters described below. In the computation for the first value of $\lambda(0)$, the algorithm begins its search for a solution at the starting point determined by the parameter start. At each successive value of $\lambda(t)$, the algorithm begins it's search at the point found in step $t - 1$.

This algorithm returns the last point computed. If the starting point is set to the centroid of the game (this is the default), and delLam > 0, then this algorithm computes the principal branch of the logistic Quantal response equilibrium. In this case taking the limit, as $\lambda$ goes to 0, the quantal response equilibrium defines a unique selection from the set of Nash equilibrium for generic normal form games. Similarly, for extensive form games, it defines a selection from the set of sequential equilibria. Therefore, in extensive form games, this algorithm can be used to compute approximations to a sequential equilibrium.

Limitations: This algorithm currently can become numerically unstable on some
problems for high values of \( \lambda \), and may generate singularity errors. Hence on some problems it may not be possible with the current implementation to obtain a sufficiently close approximation to an equilibrium.

The following parameters can be set:

- **minLam**: Sets the minimum value of \( \lambda \). Default is \( \lambda = 0.01 \).
- **maxLam**: Sets the maximum value of \( \lambda \). Default is \( \lambda = 30.0 \).
- **delLam**: The constant, \( \delta \), used in incrementing. Default is \( \delta = .01 \).
- **Plot type**: Determines the exponent, \( a \), used in incrementing \( \lambda \). Linear corresponds to setting \( a = 0 \), and logarithmic corresponds to setting \( a = 1 \). Default is logarithmic.
- **Starting Point**: Selects the rule for choosing the starting point of the search for the initial value of \( \lambda \). Default is the centroid, where all strategies are chosen with equal probability (not implemented yet.)
- **Accuracy**: Sets the accuracy desired in the computation. Default is 1.0e-8.
- **Run PXI**: A checkbox to determine whether to launch the external PXI program to plot the QRE correspondence. If this is checked, both of the following fields must also be specified.
- **PXI File**: Specifies a pathname to store the QRE correspondence in a format compatible for input to PXI, a program for graphical viewing and display of the output.
- **PXI command**: Specifies the location of the pxi program. A full pathname should be specified.

## QRE Grid

Performs a grid search to compute the complete logistic quantal response correspondence (as described in [9] for a small two-person normal form game.

The algorithm computes approximate fixed points of the correspondence for the corresponding for values of \( \lambda \) between \( \text{minLam} \) and \( \text{maxLam} \). Starting at \( \lambda = \text{minLam} \), \( \lambda \) is incremented according to the formula

\[
\lambda(t+1) = \lambda(t) + \text{delLam} \times (1/\lambda(t))^a.
\]

where \( \text{minLam} \), \( \text{maxLam} \), \( \text{delLam} \), and \( a \) are parameters described below. For each value of \( \lambda \), a grid search is done over all values of the probabilities for player 1, evaluated on a grid of mesh 'del p.' Points are evaluated in terms of the value of an objective function that measures the distance between the original point, and the best response to the best response (under the logistic best response function.) Points with a value of the objective function less than \text{Tolerance} are approximate fixed points, and are kept, others are discarded. A two stage procedure is used, in which the entire space...
is searched on a course grid of mesh 'delp1'. Any points on the course grid for which the value of the objective function is less than 'tol1' are then searched on a finer grid of mesh 'delp2' around that point. Points on the finer grid for which the value of the objective function is below 'tol2' = 'tol' are kept.

**Limitations:** This algorithm is very computationally intensive, and should only be attempted on small games.

The following parameters can be set for QRE Grid.

- **minLam:** Sets the minimum value of $\lambda$. Default is $\lambda = .01$.
- **maxLam:** Sets the maximum value of $\lambda$. Default is $\lambda = 3$.
- **delLam:** Specifies the rate at which the value of Lambda changes. Has a default value of .1.

**Plot Type:** Specifies whether to have geometric or linear incrementing. Default is 0, resulting in $a = 1$, or !geometric incrementing.

- **Grid 1 Del:** Grid size for search over course grid of probability space.
- **Grid 1 Tol:** Maximum value of objective function for which to accept solution on course grid.

**Use MultiGrid:** Checking this box causes the two stage procedure to be used, in which case the next two parameters need to also be specified.

- **Grid 2 Del:** Grid size for search over fine grid of probability space.
- **Grid 2 Tol:** Maximum value of objective function for which to accept solution on fine grid.

**Run PXI:** A checkbox to determine whether to launch the external PXI program to plot the QRE correspondence. If this is checked, both of the following fields must also be specified.

- **PXI File:** Specifies a pathname to store the QRE correspondence in a format compatible for input to PXI, a program for graphical viewing and display of the output.
- **PXI command:** Specifies the location of the pxi program. A full pathname should be specified.

**Lcp**

This algorithm formulates and solves the game as a linear complementarity problem. For a normal form game, this algorithm searches for equilibria of the specified normal form game using the Lemke-Howson algorithm, as described in [11]. Eaves[12] lexicographic rule for linear complementarity problems is used to avoid cycling.
In the Lemke Howson algorithm equilibria are found by following paths of "almost''
equilibria, where one relaxes at most one constraint. Equilibria are thus inter-connected
by networks of paths that result when different of the constraints are relaxed. One can
find the set of "accessible'' equilibria in such methods by starting at the extraneous
solution and then tracing out this entire network. See, e. g., Shapley, [13]. However, the
set of accessible equilibria is not necessarily all Nash equilibria.

For extensive form games, this algorithm implements Lemke's algorithm on the
"sequence form'' of the game, as defined by Koller, Megiddo and von Stengel, in [7].

Limitations: This algorithm is fast, but only works for two person games. [14] and [15]
have suggested ways in which the Lemke-Howson Algorithm can be extended to
general n-player games, but these extensions require methods of tracing the solution to
a set of non linear simultaneous equations, and have not been implemented in GAMBIT.
Also, on some problems with data type of Double, the current implementation can exhibit
numerical instability which in extreme cases can even lead to incorrect solutions.
Solving the same problem with Rationals will resolve any such difficulties. However, the
algorithm is much slower when operating on Rationals than on Doubles.

The following parameters can be specified:

    **Stop After:** Specifies the number of equilibria to find. If not specified, the default
value is zero, which means that all equilibria reachable by the algorithm (see
above) are to be found.

**Liap**

Finds Nash equilibria via the Lyapunov function method described in [16]. Works on
either the extensive or normal form. This algorithm casts the problem as a function
minimization problem by use of a Lyapunov function for Nash equilibria. This is a
continuously differentiable non negative function whose zeros coincide with the set of
Nash equilibria of the game. A standard descent algorithm is used to find a constrained
local minimum of the function from any given starting location. Since a local minimum
need not be a global minimum (with value 0,) the algorithm is not guaranteed to find a
Nash equilibrium from any fixed starting point. The algorithm thus incorporates the
capability of restarting. The algorithm starts from the initial starting point determined by
the parameter 'start'. If a Nash equilibrium is not found, it will keep searching from new
randomly chosen starting points until a Nash equilibrium has been found or the
maximum number of tries (parameter 'ntries') is exceeded, whichever comes first. For
an extensive form game, if the algorithm converges, it converges to a sequential
equilibrium (Andrew Solnick, personal communication).

Limitations: The disadvantages of this method are that it is generally slower than any of
the above methods, and also, there can be local minima to the Liapunov function which
are not zeros of the function. Thus the algorithm can potentially converge to a non Nash
point. However, inspection of the objective function can determine if this problem has
occurred. If the objective function is zero, a Nash equilibrium has been found. If it is
greater than zero, the point is not Nash. The algorithm will automatically check this. If
the objective function is larger than the tolerance, then the point is discarded.
The following parameters can be set;

**Stop After:** Sets the number of equilibria to find. Has a default value of 1. Note that checking **Find all** results (intentionally) in an infinite loop that will continue until terminated by the user.

**nTries:** Sets the maximum number of attempts at finding each equilibrium. Note that a value of 0 means the algorithm will continue forever, or until the algorithm is terminated by the user, whichever comes first.

**Accuracy:** Sets the accuracy desired in the computation. Default is 1.0e-8.

**start:** Sets the starting profile for the descent algorithm. **Default** is the centroid.

**Lp**

This algorithm formulates and solves the game as a linear program, and finds the minimax solution of the game. This algorithm only works for two person, zero sum games.

This algorithm only finds one Nash equilibrium. However, For a constant sum game, any other equilibria will have the same value. There are no algorithm specific parameters.

**PolEnum**

Solves for all totally mixed Nash equilibrium of the game on a given support. The algorithm iterates through all sub-supports of the initial support to find the full support equilibria on each sub-support. Supports with singular solutions are skipped by the algorithm when determined to have singular solutions.

On each sub-support, the algorithm starts with a cube containing the space of possible solutions and proceeds recursively. The recursion step begins with a subcube. The subcube is discarded if the cube is irrelevant in the sense of lying outside the space of possible solutions. Otherwise a modified Newton's method is used to search for a solution in the subcube. In the event that such a solution is found, Taylor's series information at the solution is used to inquire whether the solution is necessarily the unique solution in the subcube. If Newton's method leaves the subcube before finding a solution, Taylor's series information at the center is used to inquire whether we can be sure that the subcube contains no solutions. If neither of these procedures resolves the issue, the subcube is subdivided and this recursion is performed on each smaller subcube. Supports with singular solutions

The following optional parameters may be used to modify the behavior of the algorithm:

**Stop After:** By default, all equilibria are found. This parameter may be used to specify a maximum number of equilibria to be found.
**Singular Supps:** Returns a list of the supports which have singular solutions. **Note:**
This is currently not implemented in the GUI, but is available in the GCL.

**recurse:** Determines whether to recurse to all sub supports. Default is True.

**SimpDiv**

Computes a Nash equilibrium to a normal form game based on a simplicial subdivision algorithm. The algorithm implemented is that of [17]. The algorithm is a simplicial subdivision algorithm which can start at any point in the simplex. The algorithm starts with a given grid size, follows a path of almost completely labeled subsimplexes, and converges to a completely labeled sub-simplex that approximates the solution. Additional accuracy is obtained by refining the grid size and restarting from the previously found point. The idea is that by restarting at a close approximation to the solution, each successive increase in accuracy will yield a short path, and hence be quick.

In its pure form, the algorithm is guaranteed to find at least one mixed strategy equilibrium to any n-person game. Experience shows that the algorithm works well, and is acceptably fast for many moderate size problems. But in some examples it can be quite slow. The reason for this is that sometimes after restarting with a refined grid size, even though the starting point is a good approximation to the solution, the algorithm will go to the boundary of the simplex before converging back to a point that is close to the original starting point. When this occurs, each halving of the grid size will take twice as long to converge as the previous grid size. If a high degree of accuracy is required, or if the normal form is large, this can result in the algorithm taking a long time to converge.

In order to combat the above difficulty, a parameter 'leash' has been added to the algorithm which places a limit on the distance which the algorithm can wander from the restart point. (Setting this parameter to 0 results in no limit, and gives the pure form of the algorithm.) With this parameter set to a non trivial value, the algorithm is no longer guaranteed to converge, and setting small values of the parameter will sometimes yield lack of convergence. However, experience shows that values of the parameter on the order of 10 generally do not destroy convergence, and yield much faster convergence.

Parameters:

**Stop after:** Maximum number of equilibria to find. Default is 1.

**n Restarts:** Number of restarts. At each restart the mesh of the triangulation is halved. So this parameter determines the final mesh by the formula $1/2^{n\text{divs}}$.

**Leash:** Sets the leashlength. Default is 0, which results in no constraint, or no leash.
External Programs

One particularly useful feature of gambit is the ability to generate complete logistic quantal response correspondence for normal form games. These algorithms generate a very large amount of data that is not easily visualized. To display this data we created a specialized data plotting program \textit{PXI}.

\textbf{Figure 30: Sample PXI screenshot}

\textit{PXI} was not funded from the same source as gambit itself, but was developed largely independently by Eugene Grayver. \textit{PXI} is not limited to plotting data generated by gambit, but can be used to visualize a wide range of econometric data. It features four unique chart types, publication quality graphics output, and an easy to use graphical interface. \textit{PXI} is available on most platforms supported by gambit. We strongly recommend using \textit{PXI} if either \textit{QRE} (p. 59) or \textit{QREAll} (p. 60) algorithms are used.

\textit{PXI} is being released as \textit{shareware} and its source code is available upon request. For more information on \textit{PXI} see \url{http://www.cco.caltech.edu/~egrayver/pxi.html}
References


Glossary

Action

An action is a collection of branches, one at each node of a player’s information set, which are indistinguishable to the player making a decision at that information set.

In Figure 1, each information set has two actions. For the second player, at her information set, the two branches labeled MEET are one action, and the two branches labeled PASS are another action.

Branch

A branch is the line connecting two nodes, one of which immediately follows the other. Branches are numbered from 1 to k, where k is the number of branches at the node. Branch 1 is the uppermost branch. Branch 2 is the second uppermost branch, etc.

In Figure 1, each decision node has two branches, but in other extensive forms, the number of branches can be different.

Contingency

A contingency for player i is a profile of strategies adopted by the other n - 1 players.

Decision Node

A decision node is a node which has at least one node that follows it. Decision nodes represent points at which either a player or chance must make a decision. The different choices are represented by branches.

In Figure 1, there are a total of five decision nodes (including the chance node). The number of the player who makes a choice at a node appears underneath the node as the first number after the parenthesis. On the graphics screen, decision nodes are color coded, with each player represented by a different color.

Domination

A strategy s for player i weakly dominates strategy t if, in every contingency, s is at least as good as t, and there is some contingency in which it is better. The strategy s strongly dominates t if it is strictly better than t for player i in every contingency.

In the poker example of Figure 1 (p. 13), for player 1, strategy 21 is weakly dominated by strategy 12, and strongly dominated by 11.

GUI

This Program -- the Gambit Graphics User Interface.

GCL
The Gambit Command Language -- a programming language for manipulation and solution of extensive and normal form games. This language is designed for repetitive and computer intensive operations on games.

**Indexed Traversal Order**

This is the ordering imposed on the nodes of a game tree by a lexicographic ordering of the nodes when each node is identified by the sequence of branch numbers necessary to reach it.

**Information Set**

An information set is a collection of nodes which are all controlled by the same player, but which are indistinguishable for the player who is making the decision. Since any two nodes in the same information set are indistinguishable, they must have exactly the same number of immediately following nodes.

In Figure 1, player 1 has two information sets, labeled (1, 1) and (1, 2), and player 2 has one, labeled (2, 1). Player 2 has only one information set because Player 2 does not know whether Player 1 drew a red card or a black card from the deck. So 2's decision cannot be contingent on that information.

**Nash Equilibrium**

A Nash equilibrium is a strategy profile having the property that no player can strictly benefit from unilaterally changing its strategy, while all other players stay fixed.

**Node**

A node is either decision node or a terminal node. In the graphics display, a node is any location to which you can navigate the graphics cursor. In Figure 1, there are a total of eleven nodes -- namely five decision nodes (including the chance node), and six terminal nodes.

**Outcome**

Outcomes are payoff vectors that associate with each player a payoff. Outcomes can be attached to any node, terminal or non terminal, in the extensive form of the game. When play passes or terminates at a node with an outcome attached to it, each player accumulates the payoff which is associated to that player by the outcome at that node.

In Figure 1, there is an outcome attached to each of the terminal nodes. They are indicated by the pair of numbers at each terminal node. The first number represents the payoff to player 1 and the second the payoff to player 2.

**Poker Description**

Each player ante's $1.00 into the pot before starting
Player 1 (RED) draws a card from a deck. Player 1 observes the card, 2 (BLUE) does not.
Player 1 then decides whether to FOLD or RAISE.
If player 1 chooses FOLD, then
Player 1 wins the pot if the card is RED (a net gain of $1.00 to RED).
Player 2 wins the pot if the card is BLACK (a net loss of $1.00 to RED).

If player 1 chooses RAISE, then player 1 throws a dollar in the pot, and player 2 moves
If player 2 chooses PASS, then player 1 wins the pot (a net gain of $2.00 to RED)
If player 2 chooses MEET, player 2 throws a dollar in the pot, and 1 must show the card:
If the card is RED Player 1 wins the pot (a net gain of $2.00 to RED).
If the card is BLACK Player 2 wins the pot (a net loss of $2.00 to RED)

Pure Strategies

A pure strategy for player i is a plan of action for that player for the entire game. Thus, it is a specification of what branch to select at each of the player’s information sets. If player i’s jth information set has k(j) branches. Then the total number of pure strategies for player i is k(1) x k(2) x ... x k(J).

Realization Probability

With each strategy profile, there is associated, to each node in the extensive form, a realization probability. This is the probability of reaching that node under the given strategy profile. This probability is computed by finding the path from the root node to the given node, and then computing the product of the probabilities of selecting each branch along the path.

Note that all nodes, not just terminal nodes have realization probabilities attached to them. The realization probability of the decision node at RED RAISE is 0.5. The realization probability of the Root node is always 1.

Reduced Normal Form

The reduced normal form is the game that results when all strategies that are identical (result in the same payoffs for all contingencies of the other players are represented by a single strategy.

In many extensive form games, there are cases in which a choice adopted by a player early in the play precludes that player from ever reaching other of its information sets. In this situation, the decisions that the player makes at the unreached information sets can not possibly affect the payoff to that player (or to any other player). Two strategies for a given player that differ only in what branch is selected at an unreachable information set will generate rows (or columns, as the case may be) in the normal form that are identical. The reduced normal form eliminates such duplicated strategies, representing each duplicated strategy by its wildcard notation.

Root Node
The root node of a tree is the node with no parent. In the GAMBIT graphics representation, the Root node is always the node that is furthest to the left. It can be reached by successively pressing ←.

In Figure 1, the Root node is the unlabeled node, furthest to the left.

**Strategy Profile**

A strategy profile is an n-tuple of strategies, one for each player. The total number of strategy profiles is equal to the product of the number of strategies for each player. In the game of Figure 1, the total number of strategy profiles is 8.

**Subgame**

A subgame is a subtree with the property that for every information set containing members in the subtree, all members of the information set are elements of the subtree. In other words, every information set is either contained in or has empty intersection with the subtree.

In Figure 1, there are no proper subtrees of the extensive form, because every subtree except that starting at the Root node breaks up Player 2’s information set.

**Subtree**

A subtree is a decision node together with the collection of nodes that follow it in the tree. Note that any subtree itself has a tree structure.

In Figure 1, each decision node defines a subtree consisting of itself and its followers.

**Terminal Node**

A terminal node is a node which has no other node following it. Terminal nodes represent points at which the extensive form game ends, and typically have outcomes attached to them.

In Figure 1, there are six terminal nodes, each followed by a pair of numbers representing the payoffs to each player when that terminal node is reached.

**Topological Tree**

A topological tree (also referred to as a Game tree) is simply a collection of nodes which are connected together by branches in a way that looks like a tree: every node has at most one predecessor (parent), and exactly one node (the root node) has no predecessors. The Game tree represents a sequence of choices in the chronological order that they occur.