Introduction

1. The Sudoku problem and the resolution methods

1.1. Statement of the Sudoku problem

Given a 9x9 *grid*, partially filled with *numbers* from 1 to 9 (the "entries" of the problem, also called the "clues" or the "givens"), complete it with numbers from 1 to 9 so that in every of the nine *rows*, in every of the nine *columns* and in every of the nine disjoint *blocks* of 3x3 contiguous *cells*, the following property holds:

– there is at most one occurrence of each of these numbers.

Although this defining property can be replaced by either of the following two, that are obviously equivalent to it, we shall stick to the first formulation, for reasons that will appear later (in chapter IV, section 1.2):

– there is at least one occurrence of each of these numbers,

– there is exactly one occurrence of each of these numbers.

Since rows, columns and blocks play similar roles in the defining constraints, they will naturally appear to do so in many other places and it is convenient to introduce a word that makes no difference between them: a *unit* is either a row or a column or a block. And we say that two cells *share a unit* if they are either in the same row or in the same column or in the same block (where "or" is non exclusive). We also say that these two cells are *linked*, or that they *see* each other. It should be noticed that this (symmetric) relation between two cells, whichever of the three equivalent names it is given, does not depend in any way on the content of these cells but only on their place in the grid; it is therefore a straightforward and quasi physical notion.

As can be seen from the definition, a Sudoku grid is a special case of a Latin Square. Latin Squares must satisfy the same constraints as Sudoku, except the condition on blocks. The practical consequences of this relationship between Sudoku and Latin Squares will appear throughout this book (and the logical relationship between the two theories will be fully clarified in chapter IV).

Figure 1 below shows the standard presentations of a *problem grid* (also called a *puzzle*) and of a *solution grid* (also called a *complete Sudoku grid*).

							1	2
				3	5			
			6				7	
7						3		
			4			8		
1								
			1	2				
	8						4	
	5					6		

6	7	3	8	9	4	5	1	2
9	1	2	7	3	5	4	8	6
8	4	5	6	1	2	9	7	3
7	9	8	2	6	1	3	5	4
5	2	6	4	7	3	8	9	1
1	3	4	5	8	9	2	6	7
4	6	9	1	2	8	7	3	5
2	8	7	3	5	6	1	4	9
3	5	1	9	4	7	6	2	8

Figure 1. *A puzzle (Royle17-3) and its solution*

1.2. Resolution methods, candidates

The problem statement lists the constraints a solution grid must satisfy, i.e. it says *what* we want. It does not say anything about *how* we can obtain it: this is the job of the *resolution methods* and the *resolution rules* on which they are based (two notions that will be progressively refined in this introduction, until the final definition of a resolution rule can be given in chapter IV).

Different kinds of resolution methods can be used, depending mainly on whether they are primarily intended for a human solver or for a machine. For instance, one may be interested in efficient machine solving techniques; one will then choose machine representations of the problem specially adapted to efficient processing but that may be rather obscure for most human Sudoku players; one well-known technique of this kind relies on graph theory and maximal cliques. Although we do not neglect efficiency matters and have developed an automatised solver (SudoRules) implementing all the resolution rules described later in this book, we want to make it clear from the start that such questions are not our primary concern. Instead, our

approach is totally player oriented and we shall concentrate on formalising resolution rules that can be applied by a human solver equipped only with a sheet of paper and a pen (and quite a lot of patience) and on simulating them with the (most classical, i.e. rule based) techniques of Artificial Intelligence (AI).

	c1	*c2*	*c3*	*c4*	*c5*	*c6*	*c7*	*c8*	*c9*	
r1	3 4 5 6 8 9	3 4 6 7 9	3 4 5 6 7 8 9	7 8 9	4 7 8 9	4 7 8 9	4 5 9	1	2	*r1*
r2	2 4 6 8 9	1 2 4 6 7 9	1 2 4 6 7 8 9	2 7 8 9	3	5	4 9	6 8 9	4 6 8 9	*r2*
r3	2 3 4 5 8 9	1 2 3 4 9	1 2 3 4 5 8 9	6	1 4 8 9	1 2 4 8 9	4 5 9	7	3 4 5 8 9	*r3*
r4	7	2 4 6 9	2 4 5 6 8 9	2 5 8 9	1 5 6 8 9	1 2 6 8 9	3	2 5 6 9	1 4 5 6 9	*r4*
r5	2 3 5 6 9	2 3 6 9	2 3 5 6 9	4	1 5 6 7 9	1 2 3 6 7 9	8	2 5 6 9	1 5 6 7 9	*r5*
r6	1	2 3 4 6 9	2 3 4 5 6 8 9	2 3 5 7 8 9	5 6 7 8 9	2 3 6 7 8 9	2 4 5 7 9	2 5 6 9	4 5 6 7 9	*r6*
r7	3 4 6 9	3 4 6 7 9	3 4 6 7 9	1	2	3 4 6 7 8 9	5 7 9	3 5 8 9	3 5 7 8 9	*r7*
r8	2 3 6 9	8	1 2 3 6 7 9	3 5 7 9	5 6 7 9	3 6 7 9	1 2 5 7 9	4	1 3 5 7 9	*r8*
r9	2 3 4 9	5	1 2 3 4 7 9	3 7 8 9	4 7 8 9	3 4 7 8 9	6	2 3 8 9	1 3 7 8 9	*r9*
	c1	*c2*	*c3*	*c4*	*c5*	*c6*	*c7*	*c8*	*c9*	

Figure 2. *Grid Royle17-3 of Figure 1, with the candidates remaining after the elementary constraints have been propagated*

Given this choice, the process of solving a grid "by hand" is generally initialised by defining the "candidates" for each cell. For later formalisation, one must give a careful definition of this notion: *at any stage of the resolution process, candidates for a cell are the numbers that are not yet explicitly known to be impossible values for this cell*. At the start of the game, one possibility is to consider that any cell with

no input value admits all numbers from 1 to 9 as candidates (but more subtle initialisations can be considered).

Usually candidates for a cell are displayed in the grid as smaller and/or clearer letters in this cell (as shown in Figure 2); for better readability of such representations, the nine blocks will be marked by thick borders and each of the possible values will always be represented at the same relative place in each of the cells.

Then, the resolution process is a sequence of steps consisting of repeatedly applying "resolution rules" (some of which have become very classical and some of which may be very complex) of the general condition-action type: if some pattern (i.e. configuration) of cells, links, values and candidates for these cells is present on the grid, then carry out the action specified by the rule. Notice that any such pattern always has a purely "physical" part (which may be called its "physical" support), defined by the conditions on the cells and links between them, and an additional part, depending on the conditions put on the values and candidates in these cells.

According to the type of their action part, such rules can be classified into three categories:

– either assert the final value of a cell (when it is proven there is only one possibility left for it); there are very few rules of this type;

– or delete some candidate(s) (which we call the target values of the pattern) from some cell(s) (which we call the target cells of the pattern); as appears from a quick browsing of the available literature and as will be confirmed by this book, most resolution rules are of this type; they express specific forms of constraints propagation; their general form is: if such a pattern is present, then it is impossible for some value(s) to be in some cell(s) and the corresponding candidates must be deleted from them;

– or, for some very difficult grids, recursively make a hypothesis on the value of a cell, analyse its consequences and apply the eliminations induced by the contradictions thus discovered; techniques of this kind (named "recursive Trial and Error" or "recursive guess"), do not fit our condition-action form and are proscribed by purists (for the reason that, most of the time, they make solving the puzzle totally uninteresting); this book will show that they are very rarely needed if one admits complex chain rules.

It should be noted that all of the above resolution rules, whatever their type, do not assert that there is a solution. But for recursive Trial and Error, they may be interpreted from an operational point of view as: "from what is known in the current situation, do conclude that any solution, if there is any, will satisfy the following".

As one proceeds with resolution, candidates for each cell form a monotone decreasing set. With a little care, this remains true even when making hypotheses (i.e. resorting to recursive Trial and Error) cannot be avoided; in our "SudoRules" solver, for instance, in this case all candidates are explicitly relativised to finite sets of hypotheses (called "contexts") and monotonicity is thus maintained.

1.3. Elementary rules, Trial and Error, and their limitations

The four simpler constraints propagation rules (obviously valid) are the direct translation of the initial problem formulation into operational rules for managing candidates. We call them "the (four) elementary constraints propagation rules" (ECP):

– ECP(cell): "if a value is asserted for a cell (as is the case for the initial values), then remove all the other candidates for this cell";

– ECP(row): "if a value is asserted for a cell (as is the case for the initial values), then remove this value from the candidates for any other cell in the same row";

– ECP(col): "if a value is asserted for a cell (as is the case for the initial values), then remove this value from the candidates for any other cell in the same column";

– ECP(blk): "if a value is asserted for a cell (as is the case for the initial values), then remove this value from the candidates for any other cell in the same block".

The simpler assertion rule (also obviously valid) is called Naked-Single:

– NS: "if a cell has only one candidate left, then assert it as the only possible value of the cell".

Together with NS, the four elementary constraints propagation rules constitute "the (five) elementary rules".

A novice player may think that these five elementary rules express the whole problem and that applying them repeatedly is therefore enough to solve any puzzle. If such were the case, you'd probably never have heard of Sudoku, because it would amount to mere paper scratching. Anyway, as he gets stuck in situations where none of these rules remains applicable, he soon discovers that, except for the simplest grids, this is very far from being sufficient. The puzzle in figure 1 is a simple illustration of how you get stuck if you only know and use the five elementary rules: the resulting situation is shown in figure 2, in which none of these rules can be applied. As we shall see later (in chapter V), for this particular puzzle, there is an easy way to unblock it. But, as we shall also see, there are lots of puzzles that need rules of a much higher complexity in order to be solved. And this is why Sudoku has become

so popular: all but the easiest puzzles require a particular combination of neuron-titillating techniques and may even suggest the discovery of as yet unknown ones.

One general way out of the blocked situation described above is recursive Trial and Error: when stuck, one can start a systematic (depth first) exploration of the tree of possibilities, duly pruned by the propagation of elementary constraints (thus avoiding the exploration of obviously contradictory possibilities). Simplistic as this method may be, it has a major theoretical advantage, justifying that we keep it in our arsenal of techniques to solve a grid: it is guaranteed either to find a solution if there is (at least) one or to prove there is none.

The technical drawback is a great variance in the computation times (be it by a human or a machine), and this variability is unrelated to any sensible notion of difficulty of the grid (it depends mainly on the order chosen to explore the tree of possibilities, i.e. on chance). But the major drawback is its unrealistic character by any human standards: some puzzles would require exploring thousands of hypotheses, sometimes more than twenty levels deep. This is one of the reasons why this technique is anathemised by purists, the second being that using it generally makes the puzzle totally uninteresting.

Finally, we keep recursive Trial and Error in our arsenal, but we keep it as a last resort weapon, to be used when nothing else can be done; we shall see that with all the rules defined later in this book, such a strategy guarantees that, most of the time (in 99,7% of Royle's 36,628 reference cases defined below; in 97% of the randomly generated puzzles), this technique is not needed; and, when it is needed, we have found no case for which one level of hypothesis was not enough. With the rules for 3D chains introduced in this second edition, these percentages rise to over 99.99%.

1.4. Resolution rules and guiding principles for their formulation

Because the five elementary rules are not enough to solve any puzzle and recursive Trial and Error is not realistic from a human solver point of view, other resolution techniques must be devised.

Since Sudoku was invented, more or less complex resolution rules have been defined. They are based on the eliciting of various types of additional constraints, some of which may be non-obvious consequences of the problem statement.

Unfortunately, very often in the available literature, these rules, especially the most complex ones, are only illustrated by examples and their definitions remain

rather vague – which incurs both redundancy in the rules proposed by various authors and much uncertainty regarding their scopes of application. To check this, just look at some of the innumerable Web sites dedicated to Sudoku (more than sixty millions, only a few of which are listed in the bibliography at the end of this book).

It appears that this vagueness is due to the lack of a general guiding principle for stating the rules, and this in turn is due to the lack of a clear notion of the complexity of a rule.

Later in this book, we shall provide a precise and sometimes unusual rephrasing of most of the familiar rules. Besides the constraint of non-ambiguity, the general guiding principle we adopt can be considered a version of Occam's Razor. One can easily find some logical and some psychological support for it. It can be viewed from two complementary, but essentially equivalent, points of view:

– from the point of view of the preconditions of a rule: a rule should apply only in cases when simpler rules do not, i.e. its preconditions must be so specific as not to subsume those of simpler rules; but they must also be so general as to cover as many cases as possible; said otherwise, the scope of a rule must be extended as far as the logic underlying it allows;

– from the point of view of the conclusions of a rule (its action part): a rule should produce effective results, i.e. its conclusions should not be obtainable by simpler rules.

Of course, with the same reference to "simpler rules" in the two points of view, this principle relies on a definition of the complexity of a rule (or at least of the relative complexities of two rules). In this book, we build a hierarchy of rules progressively, based on:

– a distinction between three general classes of rules: subset rules, interaction rules and chain rules;

– a generalised notion of logical symmetry and associated representations;

– a second guiding principle: a rule obtained from another by some (generalised or not) logical symmetry must be granted the same logical complexity.

Given our objective of formalising the methods applied by a human solver, our second principle is highly debatable. There may be a great gap between abstract logical complexity and psychological complexity for the human solver. But the fact is that, in most cases, we have no idea of how psychological complexity can be measured. It is even doubtful that a given resolution rule could be given a psychological complexity measure independent of the "geographical" situation on the grid of the cells it applies to, i.e. independent of the most elementary symmetries inherent to

Sudoku (see chapter I); for instance, identical patterns of candidates on adjacent cells may be easier to see than the same patterns in distant cells; and this may also depend on individual psychological specificities. On the other hand, it is our hope that a partial relative ordering of our rules, based on their logical formulation and consistent with all the logical symmetries of the game, will serve as a reference for future measures of the psychological deviations from it. Moreover, there is a strong argument in favour of this principle, if one adopts the graphical representations and the extended Sudoku board (defined in chapter II) that makes obvious the equivalences associated to generalised symmetries.

Notice that we are looking for a partial complexity order relation on the set of resolution rules and that this is a very different task from trying to rank the puzzles based on some definition of the complexity of their resolution path (unless one defines the ranking of a puzzle as the complexity of the most complex rule necessary to solve it – not a very realistic ranking). Of course, there must be some relationship between a ranking of the puzzles and a partial complexity order on the set of resolution rules. Nevertheless, given a fixed set of rules, we shall see through examples that it can solve puzzles whose solution paths vary largely in complexity (whatever intuitive notion of complexity one adopts for the paths). In this book, we shall not tackle the problem of ranking the puzzles.

One last point can now be clarified. Everywhere in this book, a *resolution method* must be understood strictly as:

– a set of *resolution rules*,

– a *non-strict precedence ordering* among them. Non-strict means that two rules can have the same precedence (for instance, there is no reason to give a rule higher precedence than that obtained from it by transposing rows and columns or by any generalised symmetry).

As a consequence of this definition, several resolution methods can be based on the same set of rules with different partial orderings.

Moreover, to every resolution method one can associate a simple systematic procedure for solving a puzzle:

```
List the all the resolution rules in a way compatible with their precedence ordering
(i.e. among the different possibilities of doing so choose one)
Loop until a solution is found (or until it is proven there can be no solution)
|       Do until a rule applies effectively
|       |       Take the first rule not yet tried in the list
|       |       Do until its conditions pattern effectively maps to the grid
|       |       |       Try all possible mappings of the conditions pattern
```

```
|       |       End do
|       End do
|       Apply rule on selected matching pattern
End loop
```

In this context, a natural question arises: given a set of resolution rules, can different orderings lead to different puzzles being solved or unsolved? The answer is in the notion of confluence, to be explained in chapter XXII, where it will be shown that all the sets of rules introduced in this book have the *confluence property* and that the ordering of the rules is therefore irrelevant as long as we are only interested in solving puzzles; but it is of course very relevant when we also consider the efficiency of the associated method, e.g. the simplicity of the solution paths.

This abstract property has a very practical meaning for the player: it allows him/her not to be as systematic in the application of the rules as a machine would be, without running the risk of being blocked because of missing an elimination he could have done earlier in the resolution process.

2. The roles of logic and AI in this book

As its organisation shows, this book is centred on the Sudoku problem itself. Nevertheless, from the points of view of logic or AI, it can also be considered as a long exercise in either of these disciplines. So let us clarify the roles we grant them.

2.1. The role of logic

Throughout this book, the primary function of logic will be that of a compact notation tool for expressing the resolution rules in a non ambiguous way and explicating the symmetry relationships between them (the simplest and most striking example of this is the set of rules for Singles in section V.2).

For better readability, the rules we introduce will always be formulated first in plain English and their validity will only be established by elementary non-formal means. The non mathematically oriented reader should therefore not be discouraged by the logical formalism. He can even skip chapters III and IV and the formal version of each rule that will usually follow its intuitive definition.

Moreover, in the very important case of the various types of chains we shall consider, the associated rules will always be expressed in an intuitive graphical formalism (partly inspired from existing informal representations one can find on Web forums, but also resolutely diverging from them when necessary); it will be

shown to be strictly equivalent to logical formulæ – so that the explicit writing of the corresponding logical formulæ will not even be needed.

The formalism we use relies effectively on the strictest formal logic and it would not be very difficult to use it as a basis for formal proofs. From a logical point of view and given the basic definitions of chapters III and IV, we consider that these formal proofs are no more than easy exercises for students in logic and we should not overload this book with them.

As a fundamental and practical consequence of our strict logical foundations, the natural symmetry properties of the Sudoku problem can be transposed into three formal meta-theorems allowing one to deduce systematically new rules from given ones (see chapters I and IV). This will allow us to introduce chain rules of completely new types ("hidden chains").

Finally, the other role assigned to logic is that of a mediator between the intuitive formulation of the resolution rules and their implementation in our AI program (SudoRules, or any other). This is a methodological point for AI (or software engineering in general): no program development should ever be started before precise definitions of its components are given (though not necessarily in strict logical form) – a common sense principle that is very often violated, even by those who consider it as obvious (this is the teacher speaking)!

2.2. The role of AI in this book

What role do we impart to AI in this book?

The resolution of each puzzle by a human solver needs a significant amount of time. Therefore, the number of puzzles that can be tested "by hand" against any resolution method is very limited. Simulating human solvers by AI will allow us to test tens of thousands of puzzles (see section 3.1 below). This will give us indications of the relative efficiency of different rules. It is not mere chance that the writing of this book and the development of our SudoRules solver occurred in parallel. Abstract definitions of relative complexities of rules were checked against our puzzle collections for their resolution times.

Productivity of new rules was tested as soon as they were introduced. Sometimes, it was very hard to find an example for a rule (such as rules for Naked-, Hidden- or Super-Hidden- Quadruplets). And sometimes, when no example could be found, it led to the conjecture, and then to the proof, that the supposedly new rule was subsumed by (i.e. could be reduced to) simpler ones.

As I said above, this book can also be considered as a (long) exercise in AI. Many computer science departments in universities have taken Sudoku as a basis for various projects. My personal experience is that it is a most welcome topic for a project in computer science or AI.

Actually, this is how my work on Sudoku started. But, on second thoughts, I realised that there might be something original in the point of view I had developed in the meanwhile and that it might concern a wider audience of "sudoku-ka". I therefore decided to soften the mathematical content (without sacrificing its logical foundations), in the hope that the result would be of interest to the union of the three populations instead of their intersection.

3. Examples and classification results

As can be seen from a fast browsing of this book, many examples are scattered in every chapter, making nearly a third of the content. This is not only because a book on Sudoku without a lot of examples would be like a French lunch without cheese. All our examples satisfy precise functions and their choice is anything but arbitrary. We decided that each example should:

– be as short as possible,

– illustrate a precise rule,

– prove that the rule it illustrates cannot be reduced to simpler ones (in this sense, *the detailed resolution paths given for all the examples must be considered as proofs of independence theorems*),

– originate from a real puzzle (this may seem an obvious constraint, but one can find examples on the Web where a partial situation is displayed with no indication as to its origin; for instance, one can find an example of an xy-chain of length 20; but I have never seen any real puzzle whose resolution needed to consider such a long xy-chain).

3.1. The origin of our examples

All our examples rely on three large puzzle collections:

– the first, hereafter named the Royle17 collection, has been assembled by the graph theorist Gordon Royle; it consists of the 36,628 known (non essentially equivalent) minimal grids with a unique solution; in this context, a grid is called minimal if it has seventeen entries and it has a unique solution; it is termed minimal because there is no known example of a grid with less than seventeen entries and a unique solution (it might be called absolutely minimal, but, as of the writing of this book, it

has not been proven that grids with fewer than seventeen entries cannot have a unique solution); in order to avoid confusion with the broader notion of minimality defined below, we call this case 17-minimal; grid number n in this collection is always named Royle17-n;

– the second, hereafter named the Sudogen0 collection, consists of 10,000 puzzles randomly generated with the C generator suexg (see http://magictour.free.fr /suexco.txt for a description of the generation principles), with seed 0 for the random numbers generator; grid number n in this collection is always named Sudogen0-n;

– the third, hereafter named the Sudogen17 collection, consists of 10,000 puzzles randomly generated with the same software as above, but using a different seed (17); grid number n in this collection is always named Sudogen17-n.

All the puzzles in our three test databases are *minimal* in the following sense (broader than the one used by Gordon Royle, in that they may have more than seventeen entries): they have a unique solution and any puzzle obtained from them by eliminating any one of their entries has more than one solution. For the Royle-17 case, this property results from the assembling choices of the collection; for the two Sudogen cases, the property is included in the principles of the generating software.

As for the specific examples chosen in this book to illustrate our rules, most of them draw upon the Royle17 collection. Occasionally, we also take examples from the Sudogen0 and Sudogen17 collections. The main reason for preferring the Royle-17 puzzle database is that showing that there is a 17-minimal puzzle for which a rule applies is a stronger result than just showing that this rule applies to some grid with no specific property (but this is not really important for the purposes of this book). And the main reason for using also randomly generated puzzles is for not relying on biased databases when we study global classification results.

3.2. Uniform presentation of our examples

If we displayed the full trace of the resolution process of an example puzzle, it would take several pages, most of which would describe obvious or uninteresting steps. Indeed, in the worst cases, starting from a 17-minimal puzzle, there are 64 (81-17) unknown values, which makes 576 (64x9) candidates to be eliminated and 64 values to be asserted, that is 640 steps. In order to skip some of these steps, we shall use the following five conventions.

Convention 1: obvious elementary constraint propagation rules ECP(cell), ECP (row), ECP(col) and ECP(blk) will never be displayed.

Convention 2: let us define the theory (i.e. the set of rules) L1_0 as the union of all the above elementary (ECP) rules and the semi-elementary rules (Naked Singles and Hidden Singles – NS and HS) defined in chapter V. It can easily be checked that the final rules that apply to a puzzle always belong to L1_0, at least when these rules are given higher priority than more complex ones. Except in chapter V, where they are introduced, they will always be omitted from the end of the listing of the resolution path.

Convention 3: it can easily be checked that for most puzzles (due to the fact that they are minimal), the first rules that can be applied (not mentioning ECP) are semi-elementary propagation rules (NS and HS) whose direct effect is to add values. Except in chapter V, we shall replace the initial puzzle P by its "L1_0 elaboration", i.e. by the puzzle obtained by adding to P all the values asserted by these first semi-elementary rules. Of course, this puzzle is no longer minimal (but it originates in a clearly defined minimal one).

Convention 4: since, for complex puzzles, this is sometimes still not enough for concentrating the listing on the rule we want to illustrate, when necessary we shall replace the original puzzle by the one obtained after applying rules of higher complexity than the semi-elementary ones (but of lower complexity than the one the example intends to illustrate). If P is a puzzle and T is a Sudoku Resolution Theory (i.e. a set of resolution rules), the puzzle obtained from P by applying repeatedly all the rules in T until none of them can be applied and keeping only the values thus asserted (i.e. discarding any information on the candidates eliminated) is called *the T elaboration of P*. Notice that the T elaboration of P is a real puzzle (although not minimal). It includes all the consequences of T on P that can be expressed as values.

Convention 5: nevertheless, since the T elaboration of P discards information that can be expressed only in terms of candidates, when it is taken as the new starting point and it is submitted to a new resolution process using an extension T' of T with rules more complex than those in T, the first rules that apply may still be rules from T. The worst situation is when the T elaboration of P coincides with P (i.e. no new value is produced by T). Such an extreme situation seems nearly impossible when we start with a minimal P, but it is often the case that the T elaboration of P coincides with the elaboration of T by a much simpler theory than T (L1_0 for instance), i.e. all the rules in T do not produce more values than the rules in L1_0. A situation that arises rather frequently is when many candidates are deleted by T but few values are asserted; this makes the strategy described in convention 4 more or less inefficient. Conversely, the ideal case is when all results produced by the elaboration process are subsumed by the values asserted (this is of course the case when T is L1_0, but this condition is not necessary). In this case, we may be certain that the first rule applicable in T' will not be in T (still not mentio-

ning ECP). Therefore the T elaboration of P illustrates a situation in which the patterns used by the rules added to T in order to obtain T' are immediately visible (after rules from ECP have been applied). Whenever possible, our examples will be chosen from such ideal situations.

With the above conventions, only the interesting part of the resolution process of the initial minimal puzzle will be displayed. But, even with the economy resulting from the above conventions, some traces of resolution processes may be very long. Most of the time, we shall select examples with short traces, but this will not always be possible and, in order to help you keep in mind that these examples are somewhat exceptional and the possibilities are much more varied, especially for rules relative to long chains, longer examples will also be given from time to time (see for instance chapters XV or XVIII).

All our examples will respect the following uniform format (Figure 3), except that, due to page setting constraints, the figure displaying the puzzle, the introductory text and/or the listing may be inverted.

After an introductory text, explaining the purpose of the example and/or commenting on some particular point, a row of three grids is displayed: the original puzzle (always a minimal puzzle taken from one of our three collections), its indicated elaborated version and its solution. This is followed by a listing of the resolution path.

The first line of the resolution path indicates by which resolution theory T1 (L3_0 in the above example) the elaborated puzzle displayed in the second grid was obtained and (inside parenthesis) to which simpler elaboration T0 (L1 in the above example) it is equivalent. Both statements are useful: the second indicates the minimal theory T0 necessary (which is also the part of T1 effectively used) to get the elaborated version of P; the first indicates that P cannot be solved in (the stronger) T1 alone.

This first line of the resolution path also indicates in which theory T (stronger than T1) the resolution path that follows is obtained (L3_0+XY3 in the example). Most of the time, T will be of type T1+R, i.e. will be obtained from T1 by adding a single rule, R. Thus, *by showing that P can be solved in T1+R but not in T1 alone, the example proves that the R rule is not subsumed by (i.e. cannot be reduced to) the set of rules in T1*. This is a very important property, because the converse would mean that, given T1, R is useless. To express it, *we write that P belongs to [T1]+R*. Every example of this form can thus be considered as an *independence theorem*.

<Introductory text>

							3	1
				7	9			
	1	3	2					
		4				7		
			1					
5				4		6	7	
2	8							
			3					

4	7	5	6	8	2	9	3	1
3	6	1		7	9	2		
8	9	2		1	3		6	7
	1	3	2	5				6
6	2	4	9	3	8	7	1	5
	5	8	1	6		3	2	
5	3	9	8	4	1	6	7	2
2	8	6	7	9	5	1	4	3
1	4	7	3	2	6			

4	7	5	6	8	2	9	3	1
3	6	1	4	7	9	2	5	8
8	9	2	5	1	3	4	6	7
7	1	3	2	5	4	8	9	6
6	2	4	9	3	8	7	1	5
9	5	8	1	6	7	3	2	4
5	3	9	8	4	1	6	7	2
2	8	6	7	9	5	1	4	3
1	4	7	3	2	6	5	8	9

Figure x. *Puzzle Royle17-186, its L1 elaboration and its solution*

Resolution path in L3_0+XY3 for the L3_0 (or L1) elaboration of Royle17-186:
xy3-chain {n8 n5}r2c8 — {n5 n4}r3c7 — {n4 n8}r4c7 ==> r4c8 ≠ 8
… (Naked-Singles and Hidden-Singles)

Figure 3. *Uniform format of our examples*

Then comes the resolution path proper. *Each step in the resolution path is the application of a well defined resolution rule in T to the precisely decribed and purely factual situation resulting from the previous rule applications; the resolution path is thus a proof of the solution within theory T (where "proof" is meant in the strict mathematical logic sense)*.

Starting from the elaborated version of the puzzle, only the sequence of non-obvious resolution steps will be displayed. Each line in the sequence consists of the name of the rule applied, followed in order by: the description of how the condition part is satisfied (how the rule is "instantiated"), the "==>" sign, the conclusion(s) allowed by the "action" part. Details of the "nrc notation" used for the condition part will be described progressively with each rule we study. The conclusion part is always either that a candidate can be eliminated, symbolically written as here: r4c8 ≠ 8, or that a value must be asserted, written symbolically as e.g. r4c8 = 8. When the same rule instantation justifies several conclusions, they will be written on the same line, separated by commas: e.g. r4c8 ≠ 8, r5c8 ≠ 8.

The rule(s) of interest in the path will be displayed in bold characters. In the above example, there is only one step, the application of the XY3 rule to some clearly described pattern of cells and values.

The trace of a resolution path will always end with the line "… (Naked-Singles and Hidden-Singles)" or something similar to remind you of convention C2.

The above conventions present the following advantages for you reader, if you want to try the examples. First, you may skip the uninteresting parts and start from the central puzzle; it is not minimal, but it is a real puzzle. Then, most of the time, the first rule you will have to apply (after the obvious ECP) will be the one studied in the chapter of the example; and, when this is not the case, the steps you will have to apply before you reach this rule will be clearly indicated so that you can easily reproduce them until you reach the pattern of interest. Our examples are designed to help you detect these patterns but they suppose an active participation on your part: only the initial values are displayed; it is left to you to apply ECP and the other rules of the resolution path to reach the desired situation. Occasionally, the detailed situation at some point in the resolution path (i.e. all the values and candidates present at this point) will be displayed so that you can directly check the presence of the pattern under discussion, but, due to place constraints, this cannot be systematic.

Finally, note that all the traces of resolution processes given in this book were obtained with version 13 of our SudoRules solver (with some hand editing for a shorter and cleaner appearance), run in the CLIPS 6.24 environment (more on this in chapter XXI).

3.3. Classification results

The available literature on resolution rules has concentrated on isolated examples illustrating specific rules but systematic studies on large collections of puzzles are lacking. To palliate this deficiency, and as a concrete counterpart to our abstract ideas about rules classification, detailed numerical results about the number of grids solved by each type of rule will be given in chapter XXI. As a justification of these results (that would need far too many pages to be published in paper form), detailed lists of the corresponding grids are available on the author's Web pages (permanent address: http://carva.org/denis.berthier), together with lots of additional material.