Conclusion

What has been achieved

In this conclusion, I'd like first to highlight a few facets of what has been achieved in this book, from four complementary overlapping points of view.

1) ***From the point of view of the Sudoku addict***, the most striking results should be the following.

By formalising the familiar resolution rules, we have eliminated from some of them the ambiguities that plagued their usual presentations and we have clarified their scopes. For instance, we have shown that none of the two usual formulations of Triplets (or Quadruplets) – neither the "strict" nor the "comprehensive" – was the best; our reformulation shows how close Triplets (and Quadruplets) are to xy-chains but also why they cannot be reduced to them (nor to our stronger xyt- or xyzt-chains).

We have fully clarified the symmetry relationships that exist between Naked, Hidden and Super-Hidden Subset rules (the latter being classically known as X-Wing, Swordfish, Jellyfish, Squirmbag and other existent or non existent "fishy patterns"). Such relationships had already been partly mentioned on some Web pages, but never in the systematic way we have dealt with them here. As a result, we have naturally found the proper formulations for the Hidden and Super-Hidden Subset rules and we have proven that no other subset rule of such type can exist.

More generally, we have proven three meta-theorems that automatically produce new resolution rules (their Hidden or Super-Hidden counterparts) from existing ones.

With the introduction of new graphical representations of the problem in auxiliary 2D spaces (mainly row-number and column-number spaces), we have shown that the Hidden or Super-Hidden counterparts of well-known patterns (subsets or chains) defined in natural row-column space can be detected as easily as their originals, because in these new spaces they look like the originals do in the standard representation. We have thus extended the resolution power of such patterns.

We have defined a (non strict) complexity hierarchy between the resolution rules, compatible with their logical symmetry relationships.

We have proven certain theorems showing that particular resolution rules can be formally reduced to simpler ones in the above hierarchy (e.g. "xy-chains of length 3 are subsumed by Naked-Triplets plus XY-Wing").

We have evaluated the strength of each rule by the proportion of new puzzles ("new" according to the above hierarchy) its introduction allows to solve and, for the first time, such an evaluation has been based on a large collection of puzzles (more than 56,000), with detailed results available online.

We have given chain rules a major place in this book (they occupy more than half of it), because they are the main tool for dealing with hard puzzles but they remain the subject of much confusion. We have introduced a general conceptual framework (including the notion of a target not belonging to the chain) for dealing with all conceivable types of chains and we have applied it systematically to all the types we have defined. In particular, we have introduced an intuitive graphical language of patterns for specifying chains and their targets, abstracting them from any irrelevant facet (such as a link being a row or a column or a block), and we have shown that these patterns are equivalent to logical formulæ. In our framework, the confusing notions of "chain of inferences", "weak link" and "strong link" are never used; our chains are well defined patterns of candidates, cells and links.

We have chosen the simplest kind of homogeneous chains, the xy-chains, as our main type of chains, with all the other chains being intuitive generalisations of them.

We have proven that xy-chains and c-chains should have no loops (with the practical consequence that searching for these chains becomes simpler for both a human and a computer).

By pushing the underlying logic of xy-chains to its limits, we have introduced xyt-chains and shown that they are a very natural and powerful generalisation of xy-chains. We have also defined another independent generalisation, the xyz-chains, and combined it with the previous one to get the xyzt-chains.

With each type of chain in natural row-column space, we have associated two new types of chains, their hidden counterparts in row-number and column-number spaces, and, in particular, we have shown the unifying power of the hidden xy-chains. All these chains can be spotted in either of the 2D representations, using our Extended Sudoku Board.

We have also proven theorems allowing to combine our various types of homogeneous chains to build heterogeneous, more complex, ones.

In this second edition, we have generalised the above 2D chains, introduced their fully super-symmetric 3D versions (the nrc-, nrct-, nrcz- and nrczt- chains) and shown that all these types of chains can still be considered in a natural way as various generalisations of the basic xy-chains.

We have produced a multiplicity of well chosen examples proving that particular resolution rules cannot be reduced to simpler ones (in the above hierarchy).

In particular, we have proven that, using only the types of chain rules introduced in the first edition, it is necessary to consider chains of length more than thirty if we want to have a chance of solving all the randomly generated puzzles without resorting to Trial and Error or assuming the uniqueness of a solution.

In the first edition, we had exhibited a set of resolution rules (L13) based on only 2D chains that could solve 97% of the randomly generated minimal puzzles and 99,67% of the 36,628 17-minimal puzzles in the famous Royle database (without resorting to Trial and Error or to an assumption of uniqueness).

In this second edition, we have also exhibited a set of resolution rules (M5) that can solve more than 99% of the randomly generated minimal puzzles using only 3D chains of length no more than five and a set of resolution rules (M7) that can solve 99.9% of these puzzles using only 3D chains of length no more than seven. As psychologists consider that human short term memory has size seven plus or minus two, this means that a human being using these rules should be able to solve almost any puzzle without any computer assistance (but still with some patience). It should be noticed that these chains do not include subsets (Hinges or Almost Locked Sets or grouped chains), contrary to the currently popular chains (Nice Loops or Alternating Inference Chains), thus avoiding a potential source of exponential behaviour.

2) ***From the point of view of mathematical logic***, our most obvious result is the introduction of a strict formalism allowing a clear distinction between the straightforward *Sudoku Theory* (that merely expresses the constraints defining the game) and all the possible *Sudoku Resolution Theories* formulated in terms of condition-action rules (that may be put to practical use for solving puzzles). We have given a clear logical definition of what a "resolution rule" is, as opposed to any logically valid formula. With the notions of a resolution theory T and a resolution path (which is merely a proof of the solution within T, in the sense of mathematical logic), we have given a precise meaning to the widespread but as yet informal idea that one wants a "pure logic solution". This leads to both sound foundations and intuitive justifications for our resolution theories, exhibiting the following facets and consequences.

We have established a clear logical (epistemic) status for the notion of a candidate – a notion that is quasi universally introduced for stating the resolution rules but that does not pertain *a priori* to Sudoku Theory and that is usually used only from an intuitive standpoint. Moreover, we have shown that the epistemic operator that must appear in any proper formal definition of this notion can be "forgotten" in practice when we state the resolution rules and that this notion can be considered as primary, provided that we work with intuitionistic (or constructive) logic instead of standard logic (this is not a restriction in practice). Notice that this whole approach can be extended to any game that is based on techniques of progressive elimination of candidates.

We have also defined what a "resolution method" based on a resolution theory is and we have shown that all the resolution theories introduced in this book have the very important *confluence property*, allowing any ordering to be superimposed on their resolution rules without changing their overall resolution capacity. As a major practical consequence, in any software implementation of a resolution theory into a resolution method (e.g. in our SudoRules solver), we may take advantage of any convenient ordering of the rules.

The natural symmetries of the Sudoku problem have been expressed as three general meta-theorems asserting the validity of resolution rules obtained by some simple transformations of those already proven. These meta-theorems have been stated and proven both intuitively and formally. As a first example of how these meta-theorems can be used in practice, we have exhibited a precise relationship between well known (Naked and Hidden) Subset rules with what we call their Super-Hidden counterparts (the famous "fishy patterns") and we have proven some form of completeness of the set of known Subset rules. As a second example of how these meta-theorems can be used in practice, we have defined entirely new types of chain rules, hidden chains of various types, and shown their unifying power.

We have also devised a direct proof of the existence of a simple and striking relationship between Sudoku and Latin Squares: *a block-free resolution rule (i.e. a rule that does not mention blocks or squares) is valid for Sudoku if and only if it is already valid for Latin Squares*. Notice that it does not seem one can prove this result by using only the general methods one would expect to see used in such cases: either the interpolation theorem or the techniques of Gentzen's sequent calculus.

Finally, we have provided a very intuitive example of how difficult it may be to transform a theory formulated in terms of (a few and simple) constraints into a set of "production rules" (or condition-action rules). This also shows that, although the given constraints on rows and columns on the one hand and the constraint on blocks on the other hand can be formulated as axioms with independent predicates, many of the condition-action rules necessary to operationalise them do mix these predicates. This mixture is most visible in the 3D (or fully super-symmetric) chain rules.

3) ***From the point of view of Artificial Intelligence*** (AI), the following should be stressed.

Sudoku is a wonderful example for AI teachers. It has simpler rules and is more accessible for student projects than games such as chess or go, but it is much more complex and exciting than the usual examples one can find in AI textbooks (Tic-Tac-Toe, Hanoi Towers, Monkey and Bananas, Bricks World, …). It easily suggests lots of projects based on the introduction and the formalisation of new types of rules since no complete set of resolution rules is known (see below).

Sudoku provides a very good illustration of a basic software engineering principle: never start writing a program (or a knowledge base for an inference engine) unless you have a non-ambiguous specification for it. The logic of certain resolution rules is so subtle that the least deviation from it (e.g. forgetting that a target of a chain may not belong to it or that loops are not allowed in a chain) has "catastrophic" consequences (typically some puzzles being falsely claimed to have no solution). This can also be considered a nice illustration of Newell's classical distinction (introduced in [New 82]) between the *knowledge level*, here assimilated to the logical formulation, in which knowledge must be formulated in a "declarative" form independent of any processes that might be applied to it, and the *symbol level*, here assimilated to the set of rules in the CLIPS or the JESS language, in which the knowledge level is *operationalised* in a form that may depend (and does largely depend in practice) on the specificities of the knowledge representation and processing tools used to implement it. Although this book does not tackle this point, there are many ways a given resolution rule (as formulated in logic) can be implemented as a production rule in an inference engine, which are more or less related to the different ways it may be used in practice.

Sudoku is also a wonderful testbed for the inference engine chosen to run the knowledge base of resolution rules. The logic of the set of rules is so intricate that many functionalities of the inference engine are stringently tested, which is how we discovered a longstanding undocumented bug in the management of saliences in JESS. With long series of puzzles to solve, memory management can also be a problem (as it is in CLIPS).

The previous topic is related to a crucial problem of AI, both practical and epistemological: how can one be sure that the system does what it is intended to do? Although this is already a very difficult question for classical software (i.e. mainly procedural, notwithstanding the object oriented refinements), it is much worse for AI, for two main reasons. Firstly, the logic underlying a knowledge base is generally much more complex than a set of procedures is (otherwise it would probably be much better to solve the problem with procedural techniques) and secondly an inference engine is a very complex piece of software and debugging it is very difficult.

As a general result, using AI to prove theorems (although this has always been and remains one of the key subtopics of the domain) may make mathematicians and logicians very suspicious.

As a specific result, all the independence theorems that have been proven in this book rely on the correctness of the inference engine we used for our computations. They do not depend on this correctness when we assert that "theory T allows the following resolution path for puzzle P", since the validity of any particular path can easily be checked "by hand", whichever method was used to generate it. But these independence results depend on this correctness any time we state that a particular theory is not able to find a solution for some puzzle P. This is why all our computations have finally been done with CLIPS instead of JESS despite the fact that CLIPS regularly gets lost in memory management problems and computation times on long series of puzzles grow astronomical. But the fact that, due to its problem with the management of saliences[1], JESS misses some inferences on simpler rules, even though this may be infrequent, disqualifies it as a theorem prover in our case (so much so that missed inferences also vary from one version to the next). Obviously, this does not prove that CLIPS is bug free. The only certain conclusions are that, using the same knowledge base, CLIPS solved (a few) more puzzles than JESS and never returned any spurious message of type "this puzzle has no solution".

Of course, these independence theorems also rely on the correctness of the knowledge base we have developed to implement all the rules defined in this book.

---

[1] It seems that this bug has been corrected in the latest release of JESS (71b1) available as of the publication of this second edition. But we haven't carried out systematic tests.

In this respect, I would like to make three points:

– firstly, thanks to the monotonicity of the facts base (each rule can only add values or not-candidates), a confluence theorem has been proven, which guarantees that there cannot be unwanted interactions between the rules;

– secondly, each individual rule has received the same systematic treatment: it has been stated in plain English, with great care being taken for not forgetting any conditions; it has then been proven, still in plain English, in rather straightforward steps that can be checked by anybody; its formulation in multi-sorted logic (or, for the chain rules, in our equivalent graphical formalism) has been shown to be the direct transliteration of the English one; in turn, the CLIPS or the JESS formulation is the direct transliteration of the logical one, thus minimising the possibilities of discrepancies between successive steps in the development;

– thirdly, the current release of our solver (SudoRules 13) has been tested on more than 56,000 puzzles known to have a unique solution (and this produced the classification results in chapters XXI and XXIII); whereas an error in a rule that would illegitimately eliminate candidates leads very rapidly to the claim that a puzzle has no solution (this allowed the detection of a few subtle bugs in the first version of SudoRules), it is noticeable that SudoRules has not yet produced an incorrect result in the CLIPS environment.

4) ***Finally, considering the (currently very fashionable) notion of complexity***, problems that can be stated in simple terms but that need a complex solution are not anything new, although the general idea may remain obscure for the novice thinker. Among the most famous problems of this type, you have certainly heard of the four-colour problem in graph theory or Fermat's conjecture in arithmetic (now known as the "Fermat-Wiles Theorem", since it has been proven recently by Andrew Wiles, more than three centuries after its formulation by Fermat). But proofs of these theorems are not really accessible to the non-mathematician and the type of complexity hidden behind the problem statement therefore remains very abstract. With the notion of deterministic chaos, the second part of the twentieth century has uncovered a new type of complexity: some dynamical systems ruled by very simple equations may have very complex trajectories and two neighbouring points may follow quickly diverging paths – but this also remains a little mysterious if you do not have a minimum mathematical background.

On the contrary, with the popular game of Sudoku, you can get a feeling of another type of complexity, computational complexity (how this is related to the previous ones remains an interesting but very difficult question). Sudoku is known to be NP-complete, i.e., to state it very informally (and somewhat incorrectly), when

we consider grids of increasing sizes, resolution times grow faster than any deterministic polynomial algorithm. As you will never try to solve a Sudoku puzzle on a 100x100 grid (unless you have unlimited free access to a psychoanalyst), this may also remain an abstract definition. There is nevertheless a difference with the previous examples: you can already get a foretaste of the underlying complexity with the standard 9x9 puzzles (e.g. by comparing them to their homologues on 4x4 grids, the so-called Sudokids).

The Sudoku problem is defined by four very simple constraints, immediately understood by everybody in terms of "single occupancy" of a cell and of "mutual exclusion" in a row, a column or a block. For a classically formatted mind, it is therefore natural to think that any puzzle can easily be proven to have no solution or be solved by a finite set of simple operational resolution rules of the condition-action type: "in such a situation carry out such an action (assert a value or eliminate a candidate)". And this idea can only be reinforced if you consider the majority of puzzles published in newspapers. But the independence results proven in this book through a multiplicity of examples have shown that very complex resolution rules are indeed needed.

What this book has shown then, in both intuitive and logically grounded ways, is that writing a set of operational rules for solving an apparently very simple constraints propagation problem may be a very complex task. (Indeed, notwithstanding their overall complexity, the rules that have been defined in this book do not even form a complete resolution theory.) Moreover, as all the NP-complete problems are equivalent (through polynomial transformations) and some of them have lots of practical applications, such as the famous travelling salesman, dealing with the apparently futile example of Sudoku may provide intuitions on problems that seem to be unrelated.

What has been partly achieved (from the point of view of AI)

In the introduction, we said we wanted a set of rules that would simulate a human solver and that could explain each of the resolution steps. The explanations produced by SudoRules are largely illustrated by the listings given in this book; they are sufficiently explicit once you know the definitions of our rules and it would be easy work to make them still more explicit for those who do not know them; but we do not consider this as a very exciting topic. As for the solver facet, SudoRules does simulate a human solver, a particular kind of player who would try all our rules systematically (ordered according to their complexity) on all of their potential instantiations.

Is it likely that any human solver would proceed in such a systematic way? He may prefer to concentrate on a part of the puzzle and try to eliminate a candidate from a chosen cell (or group of cells). What may be missing then in our system is a "strategic" knowledge level: when should one look for such or such pattern? But I have no idea of which criteria could constitute a basis for such strategic knowledge; moreover, as far as I know, whereas there is a plethora of literature on resolution techniques (often misleadingly called strategies), nothing has ever been written on the ways they should be used, i.e. on what might legitimately be called strategies.

To say it otherwise, we do have a strategic level: searching for the different patterns in their order of increasing complexity. Notice that there is already more strategy in this than proposed by most of the books or Web pages on the subject. The question then is: can one define a better (or at least another) strategy? Well, the rules in this book (and the corresponding software SudoRules) are there; you can use them as a basis for further analysis of alternative strategies. One of the simplest ways to do so is to modify the complexity measure and the ordering we have defined on the set of rules. For instance, using psychological analyses, one could break or relax the symmetry constraints we have adopted.

What was not our purpose and has not been achieved; open questions

The first thing that has not been done in this book is a review of all the advanced rules that have been proposed, mainly on the Web (under varying names, in more or less clear formulations, with more or less defined scopes). The list would be too long and moreover it is regularly increasing. The best place to get an idea on this topic is in the recent book by Andrew C. Stuart ([STU 07]) or on the Web, in particular in the Sudoku Players Forum:
http://www.sudoku.com/forums/viewtopic.php?t=3315
(with the problem that chains are often considered as "chains of inferences" instead of patterns and they are sometimes classified according to absurd criteria).

Instead, our two main purposes in this regard were to take advantage of the symmetries of the game in a systematic way and to isolate a limited number of rule types, with rules definitions extended as far as the arguments used in their proofs allowed: this is how we introduced xyt-, xyz- and xzyt- chain rules (on the basis of xy-chain rules) and their hidden counterparts (on the basis of our general meta-theorems); this is also how this second edition introduced the 3D chain rules. Of course, we do not claim that there may not be another general type of rules that should be added to ours. For instance, if you admit uniqueness of the solution (i.e. add the axiom of uniqueness), much work remains to be done in order to clarify all the techniques that have been proposed to express it in the form of U-resolution

rules. But one of the main questions in this regard is: should we accept rules for nets or for chains of subsets? In a sense, AICs based on subsets appear to be nets when we try to re-formulate them as chains of candidates; but they are mild kinds of nets. nrczt-chains, which have approximately the same solving power as the most complex AICs, prove that including subsets (Hinges, Almost Locked Sets) in chains is not necessary. On the other hand, general tagging algorithms that can solve anything correspond to unrestricted kinds of nets and they are not of much help for answering the question: which (mild) kinds of nets should we accept?

Viewed from the methodological standpoint, more than proposing a final set of resolution rules, our purpose was to set some minimal standard in the way one should systematise the definition of rules in natural language, formalise them in logic (or in equivalent graphical representations), implement them (as rules for an inference engine or in any other formalism that can be run on a computer, e.g. as strictly defined colouring or tagging resolution techniques) and test their validity and efficiency through the treatment of large collections of examples. It is our opinion that only this complete cycle may bring some clarity into the subject.

The second thing that has not been achieved in this book is the discovery of a *complete* resolution theory that would make no uniqueness assumption and that would not use Trial and Error. Our strongest resolution theory (L16 in the first edition, M28 in this second edition) cannot solve all the minimal puzzles that have a single solution. It can solve *almost all* these puzzles, but not *all* these puzzles, and increasing the maximal length of the chains would not help. Indeed, no set of resolution rules is known that would allow to solve such exceptionally complex puzzles as Easter Monster. Some kind of nets may be necessary. Defining very complex types of nrczt-nets is very easy; defining useful but mild ones is more difficult.

Finally, another related question is: does our strongest resolution theory (L13, or its weak extension L16 or the 3D theory M28) detect all the puzzles that have no solution? We have found no example that could not be detected. But this nevertheless leaves the question open. Underlying this question, there is a more general informal one, still open: is formulating *a priori* necessary and sufficient criteria on the existence of a solution (criteria that would only bear on the entries of a puzzle) easier than finding a complete resolution theory?