Cognitive Support for Systems Engineering

Abstract
A system for reading system specifications and converting them into an active structure is described, as well as the extensions to Constraint Reasoning to allow its use on a complex dynamic construct. The resulting structure is intended to retain the entire meaning of the specification. Some of the problems that needed to be overcome are mentioned, as well as the benefits of using the same structure and methodology throughout the process. The possible effects on the Systems Engineering process are noted.

Introduction

The goals of Systems Engineering too often become mired in a clerical exercise, where the process destroys the most important aspect of the system being constructed, that it is a system of many interconnected parts working together to provide complex functionality based on interaction. Requirements Management is a good example of the curse. The complex intellectual construct of a contract or a specification is cut into small independent pieces, and much is promised with those small pieces in terms of traceability. The effort to provide this functionality convinces people that the task is under control. Meanwhile the understanding of the system aspects appears nowhere in any of these small independent descriptions, and so is not monitored or controlled.
It does not require too many lines in a specification before the inattentive user will refer only to a small area, and let slip the larger picture. Some specifications are complex, at the edge of what a human can comprehend, particularly when the system being described crosses many specialties, with the result that almost no-one understands it completely, and people make assumptions that all is well outside their area of expertise. Sometimes, the reasons for lack of comprehension are more mundane – no-one has time to read it, or did so three revisions back, and will run on that until some grievous error forces re-reading it in its entirety.
An alternative approach is to provide support through a method which does not require decomposition of the specification and damage to its meaning. We will attempt to show that the reason a system has not been proposed before for such a role is the very same problem, the desire to split the system into small pieces to make it manageable, thereby making the goal infeasible. If we build a system based on holistic principles, we can use it to assist us in understanding specifications without requiring their decomposition.

Building a Cognitive Model

When there are hundreds or thousands of individual lines in a specification, it would be too tedious to attempt to model them by hand. We don't mean a bit too tedious – the technical language used to create a specification is dense and rich, and able to describe almost anything we can imagine. In other words, untold billions of possibilities – something computers are notoriously poor in handling. To capture all the meaning would require hundreds of thousands of cognitive elements to be painstakingly connected together – many months or years of work, undone in an instant with a small textual revision which spreads everywhere.
There have been attempts to capture some of the meaning by focusing on a facet of the problem, temporal logic perhaps, or an ontology, but this brings up the difficulty of deciding what part of the meaning is necessary, and what part is irrelevant now and in the future.

It would be preferable to take a no excuses approach – whatever meaning there is in the specification, is to be captured and brought back alive [Brander and Lupu, 2008]. This forces the use of an automatic method, and a representation that is as flexible and rich as natural language, if somewhat more detailed. If the method is to be automatic, then not just a representation will do – the representation needs to be active in its own construction – it needs to be self-extensible. So why not take a Systems Engineering approach to what is required. Here are some specifications (we already know the solution, so the specifications are biased in its favor):

1 Specifications

1.1 Methodology

We will face billions of possibilities from the interaction of words, and will need some methodology to guide us. Constraint Reasoning [Cras 1993] was expressly developed to handle problems with many alternatives. It basically says – Cut hard and cut often. We can use that as a basic methodology, with the understanding that we are dealing with a dynamic problem, so Constraint Reasoning methods that work well on static problems may not be useful.

1.2 Volume

There may be 5,000 to 50,000 words in a specification – an upper limit of 50,000 words would seem reasonable for a complex and detailed specification.

1.3 Normalization

There are two possible approaches to vocabulary – a simple system can have a limited vocabulary and force users to remain within it (and become skilful in twisting it to their needs), or a system can allow a full vocabulary and infer the more complex meaning it allows. A limited vocabulary prevents subtlety of description. We will allow a full and expandable vocabulary. This permits the user to say something in a way that is natural for them, thus reducing mistakes. They can use an active verb, a passive verb, a noun, a participial phrase, a prepositional phrase, a negated form, any way that is natural within context.

The wide vocabulary opens up the possibility that different people will mean different things by the same words – we will be turning the words into a realized structure, where it is much easier to see exactly what is meant. We should expect some normalization of expression where vernacular or the egregious apostrophe error creeps in.

1.4 Dynamic Vocabulary

A specification will almost always introduce new terms to aid in the description – the system dictionary needs to be dynamically expandable as it reads the text.

1.5 Objects and Relations

It would be tedious to describe every object from scratch. A starter set will allow incremental building. We will assume a few thousand known objects, and a few thousand relations. It is often hard to distinguish between an object and a relation – ownership of a car, for example. Ownership is obviously a relation, but so, largely, is a car. If you void the assembly relation for a car, there are just a lot of spare parts, not something drivable. We shall treat relations as objects, with essentially no difference between the two, except that relations provide cross-linking of their parameters – the warp to the weft of inheritance in the tapestry that is the specification.

1.6 Names and Objects

There can be lengthy philosophical debate over the difference between names and objects, particularly where they appear in documents. We will treat names and the objects they name as usually the same thing, but having different inheritance. So "IBM" can be a large computer company, a type of computer, an acronym, a trademark, or a noun. Any of the properties may be important in the particular context.

1.7 Inheritance

It is efficient to allow objects to acquire properties by inheritance. But sometimes that is undesirable, making it necessary to disable the chain of inheritance. We shall allow multiple inheritance, partial inheritance ("part of the factory" still inherits the street address, but possibly not the properties of the office), and blocked inheritance. Blocking can be specific, when we want to block the inheritance of a particular property (a plural noun inherits the properties of a singular noun, except the singular attribute), or global, when we seek the real properties of an object, rather than the grammatical properties of its name.

1.8 Existence

An important part of a specification or contract deals with the existence of objects and relations. People happily say "Can he swim?" or "There are two ways of doing it", but become queasy when the crudities of conventional existential logic are proposed. We shall include existential logic, in a form that supports propagation of its logical states and intermingling with propositional logic at connectives – in other words, we will implement the existential logic of natural language.

1.9 Temporal Logic

A specification is usually almost free of temporal logic. It is written in the form "Thou shalt..", with the tense assumed to be future. Temporal logic intrudes when a sequential process is described. We shall allow for the possibility of temporal logic on all relations.

1.10 Anaphora

A well written specification has relatively little anaphora, but sooner or later an "it" is encountered. The "it" may refer to the component last mentioned, or to clause 6, paragraph b, or to the entire specification. Anaphora will be handled, along with the structure of the specification. That is, a model of the specification structure will be created as the specification is read, together with a model of the meanings of the sentences within the structure. This will allow references to the specification structure to have the appropriate meaning. A specification may control some aspects of itself – "Clause 6 becoming void on…". The modeled document structure must allow logical and existential control of parts of the document structure to be asserted or evaluated within the document.

1.11 Multiple Levels

A specification may talk about intent or purpose when providing guidance – a different level to whether the widget shall be colored blue. This requires one relation to control another. There should be no intrinsic limit on the number of such levels.

1.12 Disambiguation

A functional performance specification is notionally unambiguous, while not specifying in detail how something is to be done – by being deliberately vague. A system that transforms text into a structure can show where the structure is ambiguous. To do so means it is capable of detecting ambiguity in its structure,

which implies an ability to hypothesize. The ability to hypothesize is central to Constraint Reasoning. We shall require the ability to hypothesize during construction of the structure.

1.13 Reading Time

We are dealing with complex specifications, which must be read carefully by a human. Machine reading speed should be no worse than the same order as an attentive human seeking full understanding in one reading – faster would be preferable, but not necessary.

1.14 Some Other Things

The specification is a discourse, which we will assume is true. This truth value should be communicated to all parts of the discourse, unless the structure determines otherwise. It may be necessary, within the specification, to control the existence of some part of it – "if the vehicle uses disk brakes, then the following detailed specifications apply, otherwise refer to Section 3(a)". If the discourse uses multiple levels, then truth values should flow through the structure to represent that fact.
A specification may refer to other documents – standards, policies, procedures, a contract, a Statement of Work. We will gloss over for the moment the tar baby problem – touch one part and end up sticking to everything – and assume that capturing the entire specification is enough of a challenge.

2 Meeting the Specification

The specification does not seem too hard to write down. Perhaps it is not impossible to meet it. Let's start with a relation.

2.1 Relation Structure

We know that –

- A relation must be an object
- A relation connects other objects together
- There are possibly thousands of different relations
- Relations should layer without limit
- Many relations have multiple meanings

Here is a proposal for a relation structure.

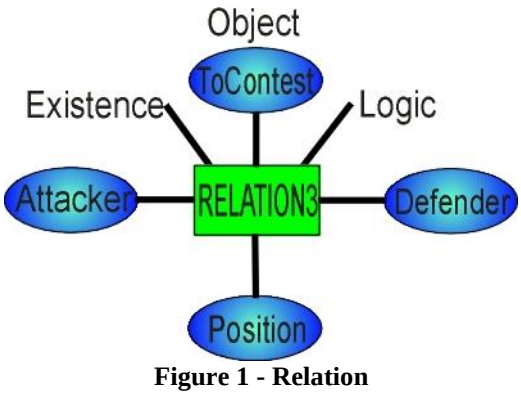

Figure 1 - Relation

It has a head, which inherits its properties, like any object. It has propositional and existential control connections, which are connected together and managed through an operator. These are connections, not inputs – logical states can flow in or out of these connections. Asserting a propositional true on the relation implies an existential

true, and asserting an existential false implies a propositional false. Intermediate Bayesian truth values are possible, but the propositional truth value cannot exceed the existential truth value, unless it is true. If there is a one percent chance that something is possible, it can be logically true, but there cannot be a seventy percent chance it will happen (in the same time window). Not shown in the figure are attributes of the relation, some of which are its time ranges for existence and validity. The operator connects the parameters of the relation in a stereotyped way – a subject and an object, and possibly a second object, depending on the type of relation. Some relations are mirrors of each other – to buy and to sell for instance, and we will need access to the mirrored information.
Some relations are more complex than the examples shown, and are dynamically created by the discourse. The entire specification can be seen as a relation, but normally there will be processes described in the specification that become new relations, and their parameters are addressed through prepositions. These relations will be treated as subnets with containing shells.

2.2 Multiple Levels
Relations can take other relations as parameters, so there is no limit to the levels that can be represented. Some relations are clausal – that is, they support clauses in the same way that the discourse supports sentences. These clausal relations pass on the discourse truth value to the clause they control. As an example

> It is intended that the widget shall be colored blue.

The relation ToIntend supports the clause by sprouting a connection from its operator to the logical state of the relation it controls, and this can occur at multiple levels –

> The contractor shall require the subcontractor to demonstrate that the machine can provide….

2.3 Heavily Packed Noun Phrases

Specifications can include some quite ugly noun phrases:

> The delivered training vehicles build state

with the system needing to unravel the noun phrase into the objects and relations that make it up – something like

> The state of build of the vehicles delivered for training

2.4 Prepositional Chains

Much of the descriptive power of a specification comes from its prepositional chains, allowing an object to be identified precisely. They are also probably the best example of why parsing of text on a purely grammatical basis is doomed to failure.

A within B down to C without D to E

A in B as C in D as E on F on G for H upon J in K by L for M to N of P for Q with R of S in T for U (this is obviously from a contract written by a lawyer)

Without knowing the meaning of the objects and the possible relations among them, these chains are impossible to unravel. We could assume each preposition operates on the object to its left, but this will fail about thirty percent of the time, and we are

aiming for 99% reliability overall. There are rules for unraveling chains which everyone picks up as they go along – one is where a preposition jumps over objects to its left, those objects then become “sheathed”, and inaccessible to other prepositions on the right. We also need to know the verb relation to the left of the chain (and may need to know the subject), as one or more of the prepositions may be a collocation with the verb, and change the verb to a different relation. When a document is as heavily laden with information as specifications are, the sentences become barely grammatical – another reason why grammar by itself is a poor basis for untangling the structure.
The preposition mechanism needs to be capable of considerable rearrangement of the sentence –

The rope shall be cut with a knife
is turned into
A knife shall be used to cut the rope.

Prepositional chains are another job for Constraint Reasoning and structural backtrack, using the meaning of the objects to guide the choice of connection. Some connections need to be made speculatively, as they will change the context in which other connections are made.

2.5 Subordinate and Superordinate Clauses

The joining together of the logicals for subordinate and superordinate clauses is another area rich in subtlety, far beyond the simple IF…THEN or “while” or “when”. Some subordinate connections (“but” is an example) cannot be done independently at the level of clausal logical control, and also require connection at the level of the relations within the clauses.

2.6 Processes

The system cycles through the following processes:

- Tagging
- Parsing
- Structure building
- Stitching together

The processes run in quasi-parallel on each sentence – building of structure will occur as soon as a noun phrase or a clause emerges from parsing, and will then suspend until another noun phrase or clause emerges. Anaphora resolution occurs when each sentence is complete and has been joined to the discourse, and the truth value has been propagated – the state of the relations will be used in the resolution.

2.6.1 Tagging

As full reading is time consuming, an initial tagging pass of the discourse is completed to throw up any obvious errors – unmatched parentheses, absent targets for clause references, etc. Then each sentence is tagged just before parsing, as any sentence may change the dictionary used in tagging of following sentences. The tagger links words in the input stream to objects in the dictionary. Safe collocations are grabbed at the word level – “in accordance with”. Some pruning occurs here – defined terms are held in a local dictionary, and recognized as such. The local dictionary is dynamic – when a new term is defined, the parser enters it in the

dictionary. All the alternatives that can be safely pruned at the word and multiword level are pruned.
Words indicating negation are removed and negation through inheritance added. If an unknown word is encountered, an external dictionary is searched, and the structure it can contribute is imported. This method works well for arcane nouns, but is not satisfactory for workhorse words like prepositions, so all these are represented initially.
The tagger also handles all the index structure of the document – Clause 2.3(a) etc. so that the parser can ignore the explicit document structure, except for references it finds within sentences.

2.6.2 Parsing

Parsing uses a constraint reasoning process – a parse chain is constructed and then flooded with sets. A further collocation pass ensues, this time with access to all of the properties of each word rather than just the word itself. Lawyerly rhetorical flourishes, such as “In no event shall”, are cut out and simpler forms inserted. Groups of objects or relations are formed into object groups, with the property that they radiate all the properties of their members. That is, “Jack and Jill” will form an objectgroup, as will “fell (down) and broke”. Sets of neighboring objects cut each other, using hierarchical ANDing. That is, a set on one side may accept a participle, on the other a present participle, so the present participle is accepted by both. A problem is that the objects may not really be neighbors, the object on the left being at the end of a relative pronoun clause and the real neighbor being far to the left, so cutting could be invalid. If portion of the chain is cut out, the pruning that has already occurred is re-evaluated – the excision is “healed”. To reduce the diversity, adverbs and free prepositional chains are cut out, and will later act as modifiers to the building process (an example of a free prepositional chain – “In operation under extreme conditions, the machine shall…”). Pattern structures recognize patterns and erect more structure, structure also flooded with sets. The process continues until the sentence level is reached. Some pattern structures will cut and rearrange the parse chain, or insert new objects – an implied subject pronoun or a clausal or relative pronoun (one reason why “that” is so pesky).
During parsing, the definition of a new term may arise. It may be in parentheses (and the parentheses be surprisingly distant from the target) and be an acronym, or use a ToDefine or ToMean or similar relation. Unfortunately, no specification is pure, and a defined term will frequently be used before it is defined – cleanup is initiated on previous sentences. The defined term may be used in the same sentence as its definition – the possibility is searched for and the mistagging cleaned up. Sometimes the definition is a distant forward reference, so parsing must continue with only the words forming the name providing information as to meaning. The addition to the dictionary is left until parsing of the sentence has completed.
The parser will often find forward or external references – in Clause 2 it finds a reference to Clause 5. There is no point moving to clause 5 to resolve the reference, as that may refer to Clause 15 or Clause 3. It builds the reference, and forward references are resolved when the new structure is built, or a list of unresolved references is left at the end of the process, indicating errors. There is a hierarchy of pending jobs – some at the end of the sentence, some at the end of the clause, some at the end of the discourse.
English, even technical English, relies heavily on inference, but the built structure cannot, so all implied objects are inserted. This may sound like a weakness – how can all implied objects be inserted before the anaphora are resolved? Where they cannot be inserted or the precise meaning identified, a more ambiguous position in the structure will be linked to. For instance, “ToTake” may have a dozen meanings in

a hierarchy. If we know nothing, linkage will occur at the least specific point. As more is learned about its parameters, the connection is moved to be increasingly specific. Constraint Reasoning has been extended, in that some patterns will change existing properties or provide new properties based on context, where providing the property without context would be damaging – like handing out hand grenades to people in the street instead of soldiers in battle. Some words are destructive of the parsing process if given all their powers at every use – “that” is a good example.

2.6.3 Structure Building

As mentioned, structure building begins as soon as a noun phrase is identified. It may be a Germanic possessive or a phrase packed with meaning, such as

“a high fidelity computer generated three dimensional virtual vehicle”

The preposition mechanism is used to unravel noun phrases and handle Germanic possessives. The result of this unraveling will then used to determine the meaning of the clause relation.
When a clause within the sentence is identified, any prepositional chains attached to the subject, verb relation and objects are built and any relation controls are gathered. Building of the meaning of a preposition may require building the clause relation, or building a different relation if the combination of verb and preposition requires it. The relation itself is built, and its control, which may be propositional or existential depending on the verb auxiliary (“shall fire”, “can fire”), is combined with others built during construction of the structure specified by the prepositional chains and noun phrases. The logicals are further combined to represent superordinate, subordinate and adversative clauses, until the sentence is fully built. The top logical is joined to a sentence logical variable and the truth value propagated.
After recovery of scaffolding (the parse chain and tree, other bits and pieces), representing the meaning of each word typically requires about ten structural elements. This gives about half a million elements for fifty thousand words (why it would be inconceivable to build such a structure by hand). An upper limit on a modern computer would be about five million structural elements or a gigabyte of memory, so specifications are well within range.

2.6.4 Stitching Together

Unknowns have been created for unspecified relation parameters, and anaphora have been identified during the building phase – “a system/the system, it, they, that entity”. These elements are resolved using the properties of the relations that connect them. Stitching together is achieved using either direct replacement of the unknown object with a known object, or synonymy is asserted using an IS operator as shown in Figure 2 (the system sits on an older system, where operators were not objects – it was more efficient and much less flexible – purely logical operators are still handled this way). To speed up searching, relations which assert synonymy – ToBe, some meanings of ToName, ToCall, ToReference – have an IS operator built in parallel, with its logical control under the same control as that of the relation.

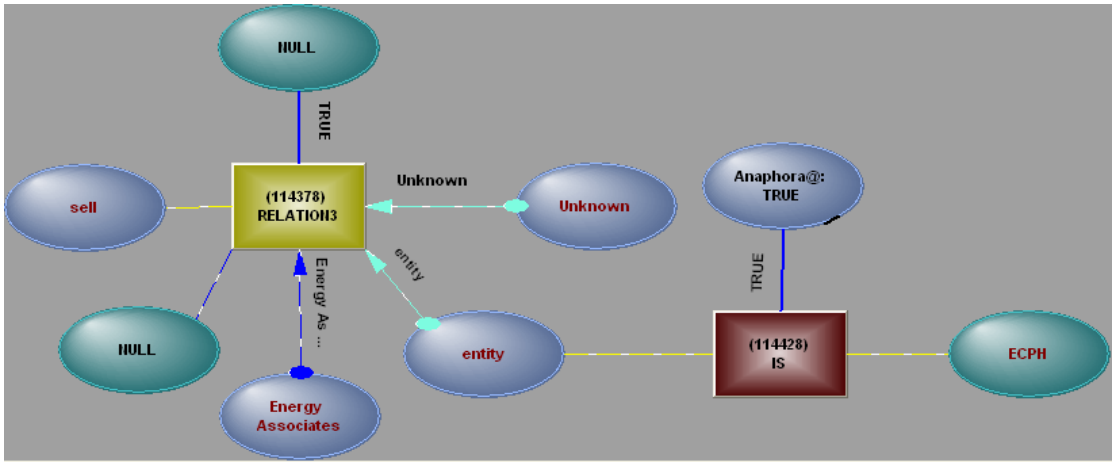

Figure 2 - Asserting synonymy

The stitching together of objects and relations, often at long range in the text, is the crucial step in developing meaning within the structure. The ability to hypothetically "plug" a possible object into the structure and observe its consistency with its connections allows the system to resolve anaphora in a more powerful way than running a few simple pronoun rules on recently mentioned objects. It is another example of the use of a dynamic form of Constraint Reasoning within the system.

2.7 Searching

A query can be posed in free text, using multiple sentences and anaphora. The query is built as a structure, linked either directly though existing objects such as defined terms, or indirectly through inheritance. The searching process identifies the minimum size starting point. Usually an explosion of possibilities will occur, where one relation generates many possibilities and one or more relations drastically prune them. The sets flow backward and forward in the query structure until no further pruning occurs. This is an excellent way of being reminded that there is something on page seventeen that is relevant.
In searching through the structure, different constructions having the same meaning need to be identified. Direct synonyms are passed through routinely, as well as antonyms requiring a reversal of logical state – to exclude, to not include. The parameters on a relation can be mapped differently, the logical states may be different – "the contractor mentioned it in the document", "the document mentioned it". Maps are used to allow traversal between relations with mappable synonymical meanings, resulting in a very dense structure, a tiny fragment being shown in Figure 3 (this part of the initial structure is being hand-built until we can work out how to automate it).

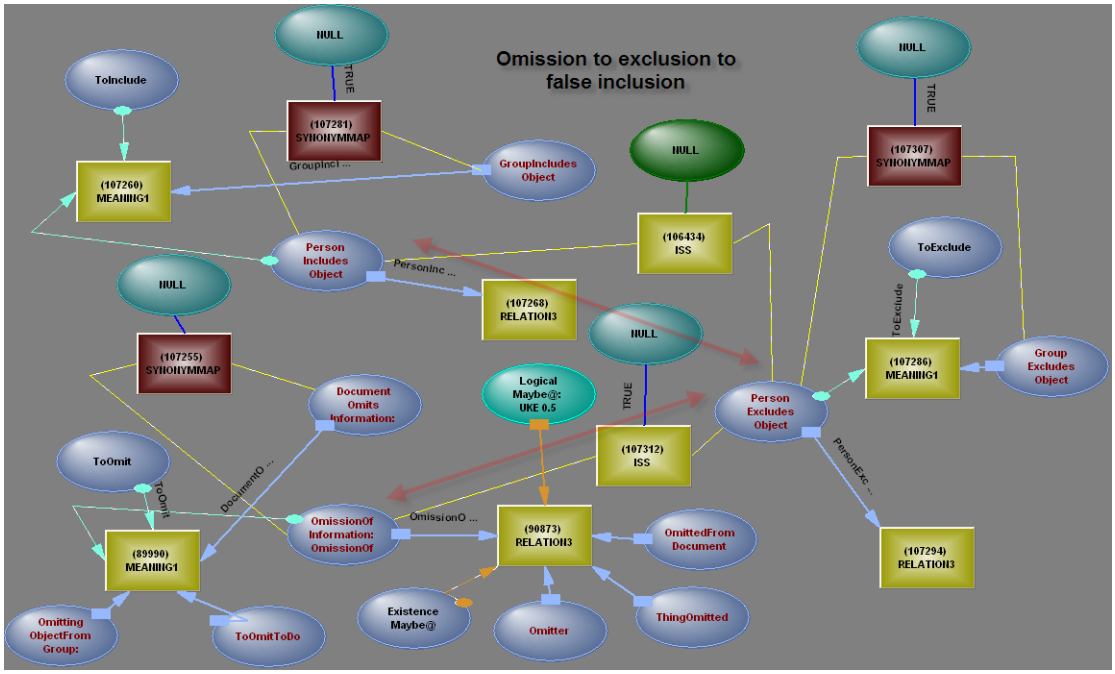

Figure 3 - Providing mapping between synonymous relations

2.8 Reading Time

The time to read a specification is similar to that of an attentive human. The system traverses its hierarchical parent structure millions of times in the reading process, with hierarchical ANDing of large sets being particularly expensive.

3 Self Extensibility

There are two possible approaches to having a system read text – either the system knows everything before it starts reading, and what it reads goes down pre-existing pathways, or what it reads creates new pathways. Pre-existing pathways may work for a trivial application, but not for a specification that describes something that has never existed in the world before. If what the system reads creates new pathways, then the system is self-extensible.

One could argue that a compiler and linker of programs can create a program that never existed before, but a compiler works on a very limited set of instructions, and assembles prewritten code fragments without any sense of change of repertoire, or looking at what it has created to help it to decide what to do next – it is static.

To require a system to be self-extensible is to make a very strong statement about how such a system must be constructed. Unless there are two modes of operation, it means that everything must be consistent in its construction from its root, so new structure blends seamlessly with the old. The structure we have described can be self-extensible, as it is made of the same elements from its root – variables, operators and links. Each of these is a structural element, and their connection together determines the operation of the system. The method of hypothesizing employs structural backtrack, so structures can be created, tested and then undone. The concept of self-extensibility is modeled on the Jacquard loom and is shown in Figure 4.

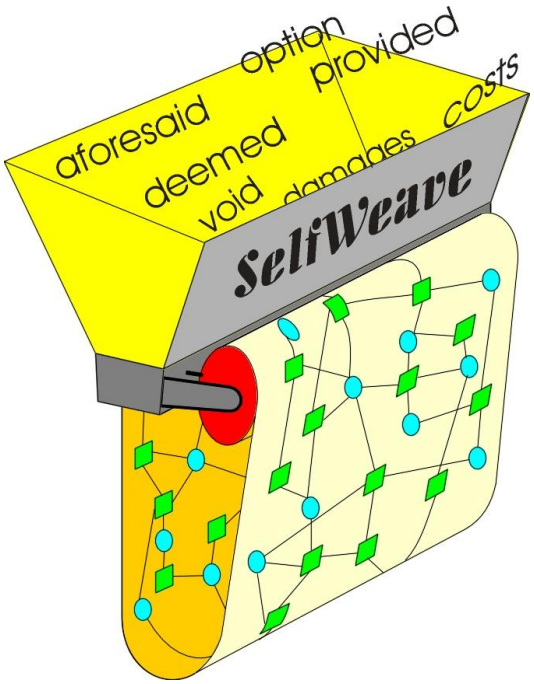

Figure 4 - A self-extensible loom

The incoming words are woven into an extension of the initial structure, and the new total structure is used to weave more structure from more words. The Jacquard loom provided a model of a system that wove a structure from a program. In the ensuing century, we attempted to create systems that could write a program from a program. The alternative, of weaving a structure from a structure, was not explored. A specification is a sophisticated and intricate structure, and we wish to create another, more active and more useful, structure from it. The same thing that drove the creation of the Jacquard loom, the need to reduce the labor in creating an intricate structure, has also driven the creation of the system being described. The parallels are even closer – a specification is created using a single thread, weaving backwards and forwards, bounding a large area, and then going back and filling in the details, much the same as a tapestry, and only when the tapestry is completed can its meaning be seen and understood.

4 System Utility

Machine reading of a specification provides utility at three points in the Systems Engineering process.

- It highlights duplication, inconsistencies and ambiguities during the generation phase of the specification.
- It provides a consistent interpretation of the specification to all interested parties. There can't be arguments about what was meant, because the meaning is directly visible (that doesn't mean the meaning can't be wrong, just that it should be much clearer what the meaning is intended to be).
- It allows conceptual and detailed comparison with proposed changes or extensions.

5 Conclusion

A system which can provide support for requirements management and analysis has been described. The same active structure is used for parsing the specification, building its meaning, the resolution of anaphora and the searching of itself. We are saying that this is not a choice available to a designer, but that there is no other way

of doing it successfully. The single structure avoids the necessity for decomposition of a specification, instead allowing the full meaning of the specification to be preserved and used in more sophisticated analysis. The resulting structure can provide much needed cognitive support for Systems Engineering as we build ever more complex systems.