Simplification of Rules Extracted from Neural Networks

HL Viktor I Cloete
{hlv, ian}@cs.sun.ac.za

Computer Science Department, University of Stellenbosch
Stellenbosch 7600, South Africa

Extended Abstract

Artificial neural networks (ANNs) have been proven to be successful general machine learning techniques for, amongst others, pattern recognition and classification. Real-world problems in agriculture (soybean, tea), medicine (cancer, cardiology, mammograms) and finance (credit rating, stock market) are successfully solved using ANNs.

ANNs model biological neural systems. A biological neural system consists of neurons interconnected through neural synapses. These neurons serve as information processing units. Synapses carry information to the neurons, which then processes or responds to the data by sending a signal to the next level of neurons. Information is strengthened or lessened according to the sign and magnitude of the weight associated with the connection.

An ANN consists of cell-like entities called *units* (also called artificial neurons) and weighted connections between these units referred to as *links*. ANNs can be viewed as a directed graph with weighted connections. An unit belongs to one of three groups: input, hidden or output. Input units receive the initial training patterns, which consist of input attributes and the associated target attributes, from the environment. Hidden units do not interact with the environment whereas output units presents the results to the environment. Hidden and output units compute an output a_i which is a function f of the sum of its input weights w_j multiplied by the output x_j of the units j in the preceding layer, together with a bias term θ_i that acts as a threshold for the unit. The output a_i for unit i with n input units is calculated as $a_i = f(\sum_{j=1}^{n} x_j w_j - \theta_i)$. Training of the ANN is done by adapting the weight values for each unit via a gradient search. Given a set of input-target pairs, the ANN learns the functional relationship between the input and the target.

A serious drawback of the neural network approach is the difficulty to determine *why* a particular conclusion was reached. This is due to the inherit 'black box' nature of the neural network approach. Neural networks rely on 'raw' training data to learn the relationships between the initial inputs and target outputs. Knowledge is encoded in a set of numeric weights and biases. Although this data driven aspect of neural networks

allows easy adjustments when change of environment or events occur, it is difficult to interpret numeric weights, making it difficult for humans to understand.

Concepts represent by symbolic learning algorithms are intuitive and therefore easily understood by humans [Wnek 1994]. One approach to understanding the representations formed by neural networks is to extract such symbolic rules from networks. Over the last few years, a number of rule extraction methods have been reported [Craven 1993, Fu 1994]. There are some general assumptions that these algorithms adhere to. The *first* assumption that most rule extraction algorithms make, is that non-input units are either maximally active (activation near 1) or inactive (activation near 0). This Boolean valued activation is approximated by using the standard logistic activation function $f(x) = 1/(1 + e^{-sx})$ and setting $s \geq 5.0$. The use of the above function parameters guarantees that non-input units always have non-negative activations in the range [0,1]. The *second* underlying premise of rule extraction is that each hidden and output unit implements a symbolic rule. The concept associated with each unit is the consequent of the rule, and certain subsets of the input units represent the antecedent of the rule. Rule extraction algorithms search for those combinations of input values to a particular hidden or output unit that results in it having an optimal (near-one) activation. Here, rule extraction methods exploit a very basic principle of biological neural networks. That is, if the sum of its weighted inputs exceeds a certain threshold, then the biological neuron fires [Fu 1994]. This condition is satisfied when the sum of the weighted inputs exceeds the bias, where $(\sum_{j/x_j \approx 1}^{n} w_j > \theta_i)$.

It has been shown that most concepts described by humans usually can be expressed as production rules in disjunctive normal form (DNF) notation. Rules expressed in this notation are therefore highly comprehensible and intuitive. In addition, the number of production rules may be reduced and the structure thereof simplified by using propositional logic.

A method that extracts production rules in DNF is presented [Viktor 1995]. The basic idea of the method is the use of equivalence classes. Similarly weighted links are grouped into a cluster, the assumption being that individual weights do not have unique importance. Clustering considerably reduces the combinatorics of the method as opposed to previously reported approaches.

Since the rules are in a logically manipulatable form, significant simplifications in the structure thereof can be obtained, yielding a highly reduced and comprehensible set of rules. Experimental results have shown that the accuracy of the extracted rules compare favourably with the CN2 [Clark 1989] and C4.5 [Quinlan 1993] symbolic rule extraction methods. The extracted rules are highly comprehensible and similar to those extracted by traditional symbolic methods.

Keywords: Rule extraction, neural networks, machine learning, comprehensibility.

Computing Review Categories: I.2.1, I.2.6.

References

[Clark 1989] P Clark and T Niblett. 1989. The CN2 Induction Algorithm, in *Machine Learning*, 3(4):216-277.

[Craven 1993] MW Craven and JW Shavlik. 1993. Learning Symbolic Rules using Artificial Neural Networks, in *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, Morgan Kaufmann.

[Fu 1994] LM Fu. 1994. Rule Generation from Neural Networks, *IEEE Transactions on Systems, Man and Cybernetics*, 24(8):1114-1124.

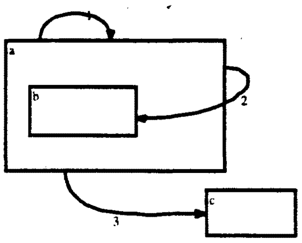
[Quinlan 1993] JR Quinlan. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufman Publishers, San Mateo, California.

[Viktor 1995] HL Viktor and I Cloete. 1995. Extracting DNF Rules from Neural Networks, *International Workshop on Neural Networks (IWANN'95)*, Torremolimos, Spain.

[Wnek 1994] J Wnek and RS Michalski. 1994. Comparing Symbolic and Subsymbolic Learning: Three Studies, in *Machine Learning*, edited by RS Michalski and G Tecuci, Morgan Kaufmann, 4:489-519.

Higraphs: an Overview of Theory and Application

Grant Hillebrand
Information Technology Advisor
Information Technology Department
Eskom
a52506@orion.eskom.co.za
P.O.Box 1091, Johannesburg, 2000



Higraphs - An Overview

ABSTRACT

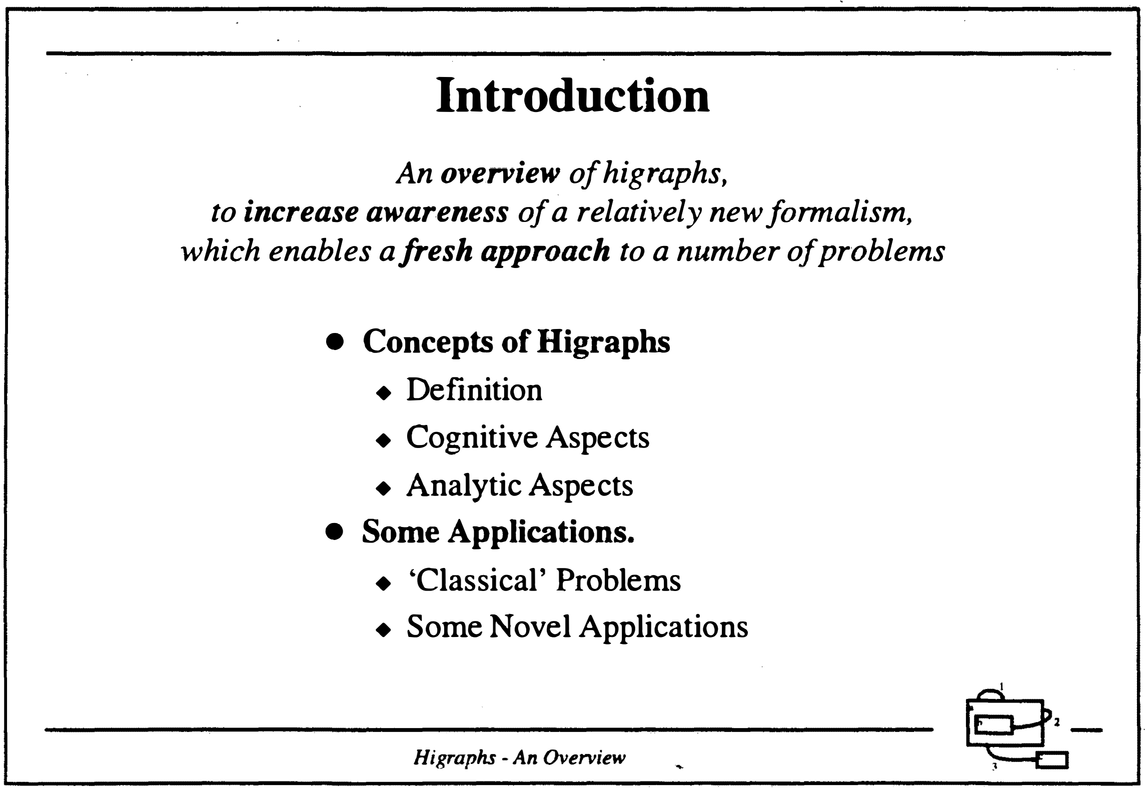
This paper presents an overview of the established concepts of David Harel's higraphs, to increase their visibility.. Higraphs are a union of extended graph and extended set theory which allows the understandable definition of complex semantics, having a powerful intuitive cognitive nature.

Viewing 'the big picture' is cited as an example of this understandability. A number of other applications of higraphs are given.

Some novel applications are suggested, including the use of higraphs in analyzing business processes, graphical user interface specification, graph domain specification and executable graphs.

A proposition is made that process graphs and data-entity state-transition higraphs are duals.

Finally, a case is made for 'informal' higraphs in group communications.



Introduction

*An **overview** of higraphs,*
*to **increase awareness** of a relatively new formalism,*
*which enables a **fresh approach** to a number of problems*

- **Concepts of Higraphs**
  - Definition
  - Cognitive Aspects
  - Analytic Aspects
- **Some Applications.**
  - 'Classical' Problems
  - Some Novel Applications

Higraphs - An Overview

This presentation is heavily based on the work of David Harel, to whom I am indebted for a very useful tool in modeling many 'real-world' situations, as well as input on their further application.

I am not a mathematician, in a very rigorous sense - although by comparison to many of my colleagues in the workplace, more so than most.

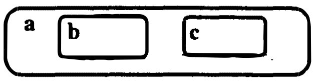
The reason for bringing higraphs to this symposium is to 'spread the word', and possibly interest some more people with various formal backgrounds in developing the application, semantics, and algorithms of higraphs.

This paper represents very much 'lessons from the field'. My aim is to focus on the concept of a powerful '*visual* formalism', rather than the rigorous *algebraic* formalism that underpins it. It is an approach often taken by engineers- "Someone has proved it, the proof makes sense, I'll get on with it and use the result"

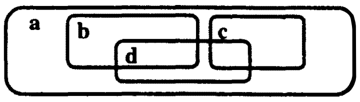
The nature of higraphs: extended sets

*The graph 'node' becomes a set element, or a '**blob**'*

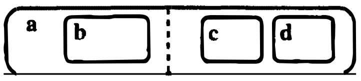
- **Classical sets:** ***containment*** **or** ***hierarchy***

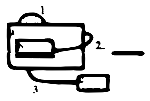


- **Classical sets:** ***'intersection'*** **or** ***'multiple states/ inheritance'***



- **'Orthogonality' of sets, 'AND' / 'OR', or** ***concurrency***





The classical graph concept of a node is generalized to 'blobs' (Harel's suggestion). The original suggestion was of rounded rectangles, but other fundamentally rectangular shapes have all been useful, particularly in extending the semantics to representing different sorts of sets.

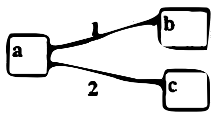
The semantics of 'inheritance' and 'concurrency' need to be defined depending on the nature of the problem to hand.

Harel originally uses the dotted line to indicate orthogonality in a classical 'AND' sense. Thus the third set above represents *{a}* decomposed in three components, valid groupings being *{b,c}* or *{b,d}*, there being a default OR implied between *{c}* and *{d}*, and an AND between *{b}* and *{c,d}*.

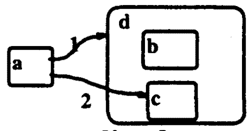
There are instances when 'OR' being the orthogonal meaning implied by a dotted line may be more intuitive. Business process modeling is one such area. The 'default' meaning of *{c,d}* is then not *{c}* OR *{d}* (i.e. one or the other) but *{c}* AND *{d}* (i.e. both simultaneously). The orthogonal operator then implies 'OR'.

The nature of higraphs: extended graphs

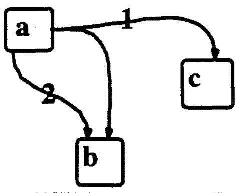
- **Classical Edges: Relations between nodes**



- **Higraph Edges: Relations between sets of blobs**



- **Binary edges generalized to *n-ary* edges**

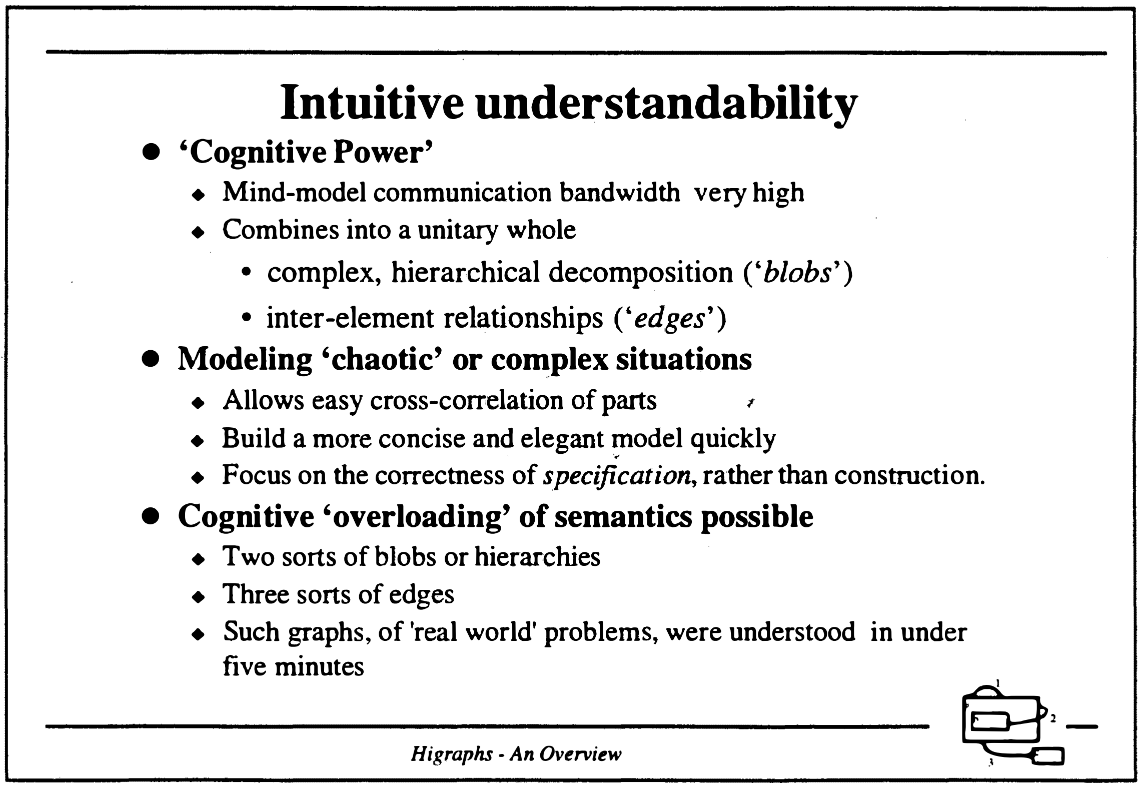


The notion of edge is generalized from the classical graph binary relationship between to edges.

The generalization of edges is not limited to directed edges only, although that is most common.

The relationship between sets of blobs is now inferred. In the second example, the relationship of edge 2 to blob *d* is defined by the specific semantics associated with the application of the higraph- it may imply a relationship or not. An example would be the lower level node's edges 'override' the 'default' specified by such a broadcast.

The generalization to *n-ary* edges, or *hyperedges* again requires its own specific semantics in each case. In general, this area has not been extensively explored, but has potential in decreasing the cognitive complexity of models of real world problems.



Intuitive understandability

- **'Cognitive Power'**
  - Mind-model communication bandwidth very high
  - Combines into a unitary whole
    - complex, hierarchical decomposition ('*blobs*')
    - inter-element relationships ('*edges*')
- **Modeling 'chaotic' or complex situations**
  - Allows easy cross-correlation of parts
  - Build a more concise and elegant model quickly
  - Focus on the correctness of *specification*, rather than construction.
- **Cognitive 'overloading' of semantics possible**
  - Two sorts of blobs or hierarchies
  - Three sorts of edges
  - Such graphs, of 'real world' problems, were understood in under five minutes

Higraphs - An Overview

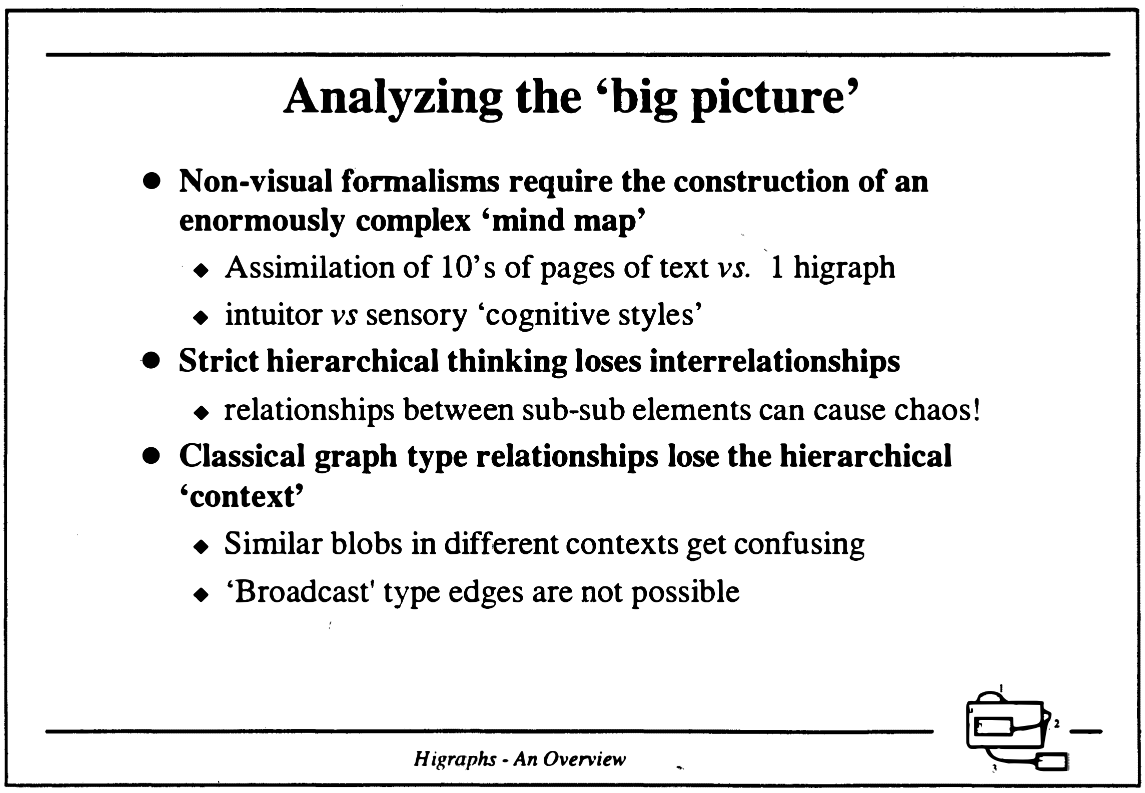
It has been proposed that 90% of the cerebral cortex is devoted to processing visual information [5]. Given this, utilizing this mode of processing most effectively clearly optimizes the communication 'band width' between the model and the human brain. The focus is on the cognitive process, enabling the mind to more effectively and completely assimilate the information to hand.

By having both decomposition and relationship visible simultaneously, there is a far better understanding than that communicated by the equivalent two 'classical' models.

This is a highly intuitive formalism. With absolutely no background to graph or set theory, people understand the content of the model very quickly. People exposed to models for the first time have been arguing about the content of the model (how it represents their business) in under 5 minutes.

'Cognitive overloading' implies different sorts of blobs and edges are quickly and easily identified by the mind. This allows overlaying different sorts of hierarchies. In business process modeling , functional groupings (organizational structure) and process elements (value chain elements), where overlaid to quickly and clearly indicate the relationship between the two.

Different edge types allow different sorts of inter-blob relationships to be specified- in our case, business events, information flows and physical product flows; in object models, messages and function calls. would be appropriate.



Analyzing the 'big picture'

- **Non-visual formalisms require the construction of an enormously complex 'mind map'**
  - Assimilation of 10's of pages of text *vs.* 1 higraph
  - intuitor *vs* sensory 'cognitive styles'
- **Strict hierarchical thinking loses interrelationships**
  - relationships between sub-sub elements can cause chaos!
- **Classical graph type relationships lose the hierarchical 'context'**
  - Similar blobs in different contexts get confusing
  - 'Broadcast' type edges are not possible

Higraphs - An Overview

Non-visual formalisms- textual or algebraic- normally remain unassociated in the human mind even when they refer to the same information. Building an association to properly understand the problem normally requires a large mental effort. It is in this area that higraphs have tremendous potential.

It has been shown that most people (75%) are 'sensory', in the Meyers-Briggs sense, implying a preference for something they can see. 25% are intuitors, preferring a internal mind map [1]. Clearly, presenting information in a more 'natural' way greatly facilitates understanding. Higraphs do this, whilst maintaining a theoretical rigor.

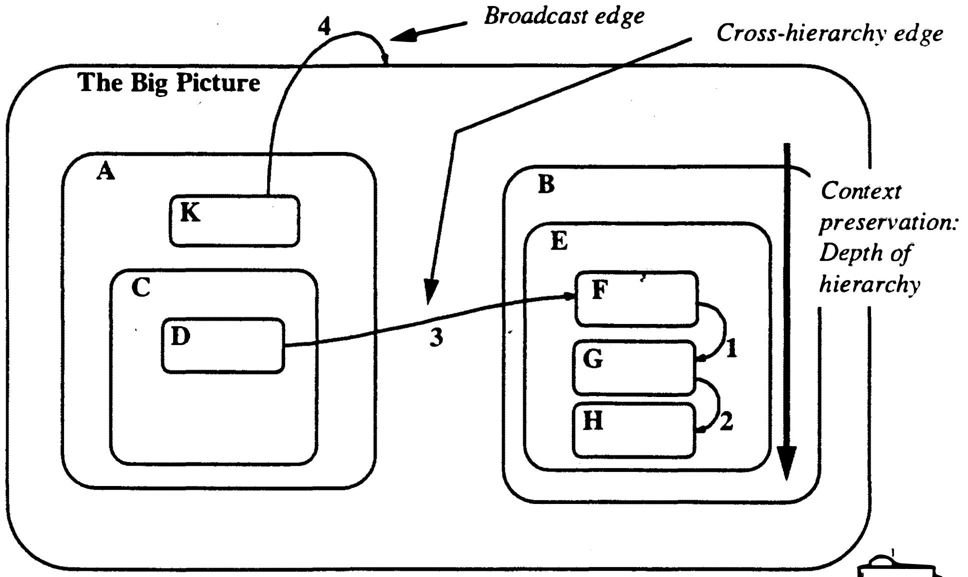
The unitary higraph delivers far more information to the mind in a much more concise fashion than a set of 'flat' or classical graphs and a hierarchical breakdown of the elements of those same graphs. Thus it is possible to simultaneously assimilate both problem context and detail, and the relationships between the two.

Typically, understanding is lost by focusing down hierarchical 'pillars', and missing the relationships across those pillars, or across some connection graph, which loses the hierarchical context of the elements. By the nature of a higraph, this loss of understanding is limited.

Since the nature of analysis and modeling tends to be inward, rather than outward, something to hold the whole together is rare.

An example follows illustrating some of these notions.

A sample 'big picture'



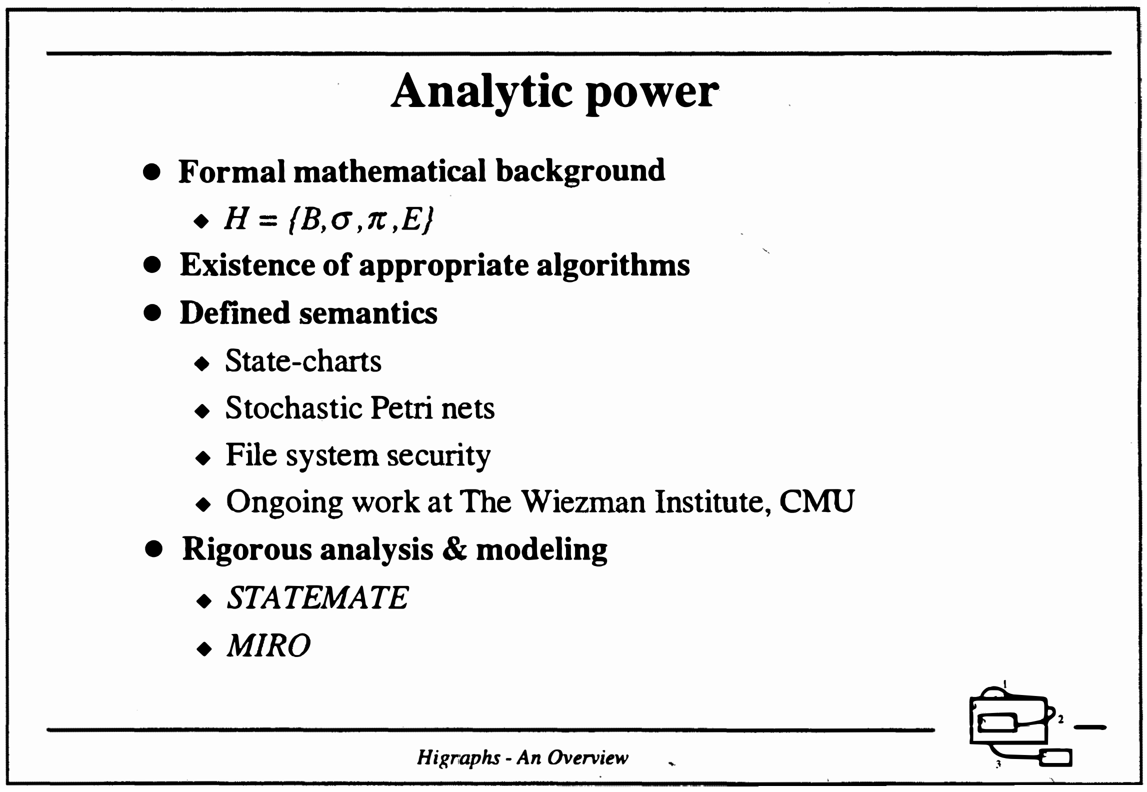
Higraphs - An Overview

The ability of an edge to relate to a whole set of blobs gives a 'broadcast' type facility. To duplicate such an edge in a classical graph would require *n(B)* edges, and be very confusing.

The preservation of context (*'The Big Picture'*) and the detail, for example the edges *1* and *2* between *F,G* and *H*, is immediately apparent to the mind, but very difficult to express without resorting to a very long-winded narrative, or complex algebra.

A danger here is trying to put *everything* into the same picture. Experience suggests an upper limit of about 80 blobs and 80 or so edges. This limit is very much dependent on the nature of the graph. If there is heavy nesting, up to 120 blobs can be shown. If the edges are 'short', in the visual sense of joining topologically close blobs, then more are possible. If they are very long, then fewer.

For large pictures (100+ blobs, 100 + edges), a repository based approach is strongly indicated, allowing the building up of separate views of the same fundamental objects.



Analytic power

- **Formal mathematical background**
  - $H = \{B, \sigma, \pi, E\}$
- **Existence of appropriate algorithms**
- **Defined semantics**
  - State-charts
  - Stochastic Petri nets
  - File system security
  - Ongoing work at The Wiezman Institute, CMU
- **Rigorous analysis & modeling**
  - *STATEMATE*
  - *MIRO*

Higraphs - An Overview

A higraph is formally defined as the set

$H = \{B$ the set of blobs,

σ the sub-blob function, building the hierarchy

π, the partitioning function, building the orthogonal partitions,

E the set of edges (on BxB for binary edges)

$\}$

Based on this definition, various semantics and algorithms have been defined and proven in various specific areas. These include, amongst others, state-charts, performance modeling and file system security.

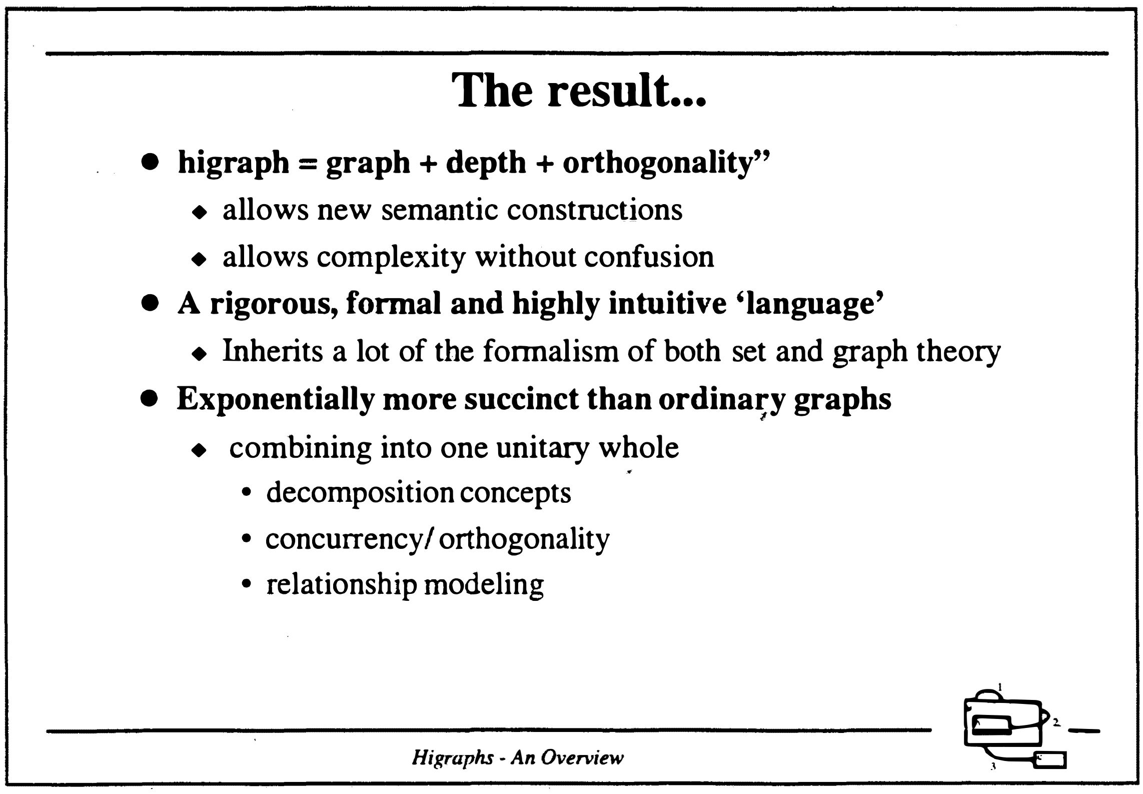
Stochastic Petri nets have been used in performance modeling (timing), looking at guaranteed execution of operating system type processes [6]

File system security processing has focused on relating sets of people to sets of files with well defined access rules [5]. The semantics allowed proving the security of the defined schema.

There is also ongoing work on algorithmics at The Wiezman Institute, defining more general, but precise semantics, allowing general graph algorithms (shortest path, etc.), to be defined for higraphs.

STATEMATE [4] and *MIRO* [5] are systems implementing the ideas of state charts and file system security respectively, with appropriately defined semantics and algorithms.

The formal analysis of state chart reachability graphs in *STATEMATE* uncovered a previously unknown path to firing a missile [4]



The result...

- **higraph = graph + depth + orthogonality"**
  - allows new semantic constructions
  - allows complexity without confusion
- **A rigorous, formal and highly intuitive 'language'**
  - Inherits a lot of the formalism of both set and graph theory
- **Exponentially more succinct than ordinary graphs**
  - combining into one unitary whole
    - decomposition concepts
    - concurrency/ orthogonality
    - relationship modeling

Higraphs - An Overview

higraph = graph + depth + orthogonality is Harel's description of higraphs.

The addition of depth and orthogonality to graphs allows a greater semantic complexity to be rigorously and understandably represented than is the case with graphs alone. This means that a whole new realm of problems are effectively addressable, by building the appropriate semantics. This is primarily as a result of making the relationship between complexity and semantic power (measured in number of nodes and edges required to express something) exponentially more succinct.

A typical example of this is the 'broadcast' edge mentioned above.

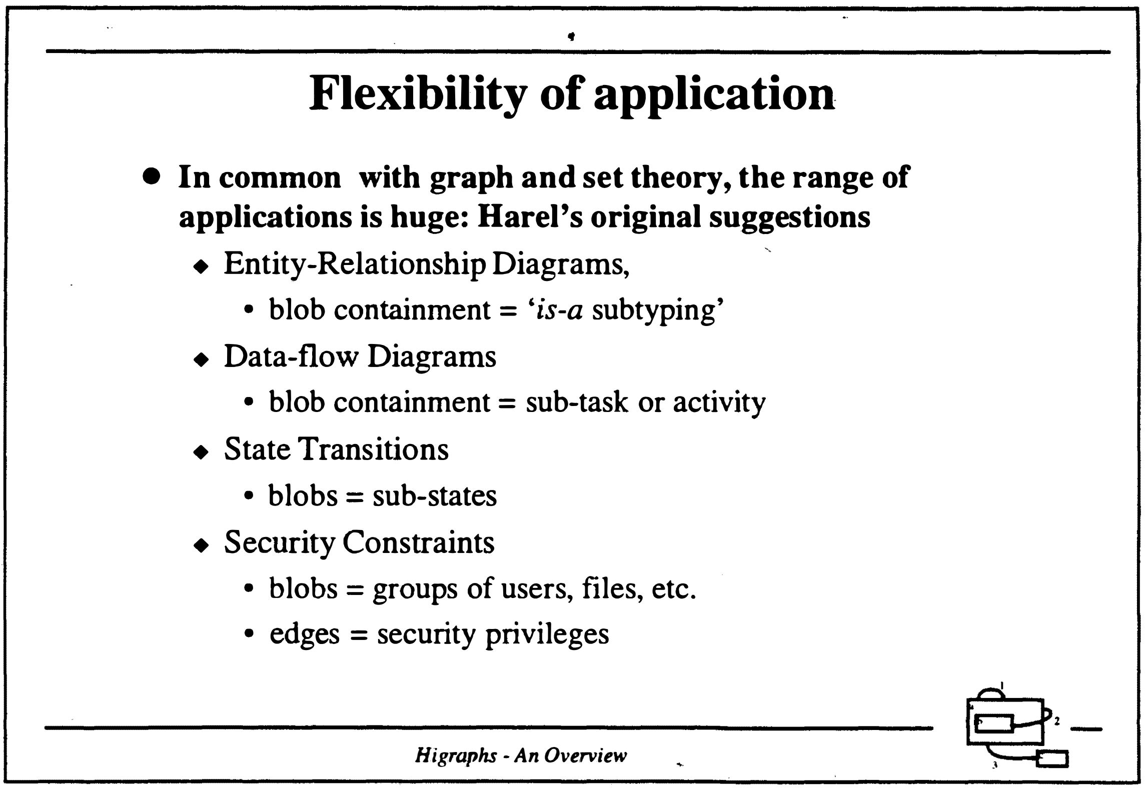
The ability to inherit both graph and set theory provides an 'instant' richness from which much higraph development can spring.

The cost of the exponential succinctness may be at the expense of computability, but it is easier to throw more compute power at a problem than to increase an individuals' brainpower!

Relationship modeling refers to the formal sense of graph binary relations, but also, more generally, any two sets of things that may be related, as will be shown in the applications that follow.

It has been our experience that associating a small amount of defining or explanatory text with each of the higraph elements, in a CASE tool environment, enhances them significantly.

In the practical examples that follow, there has normally been such text associated.



Flexibility of application

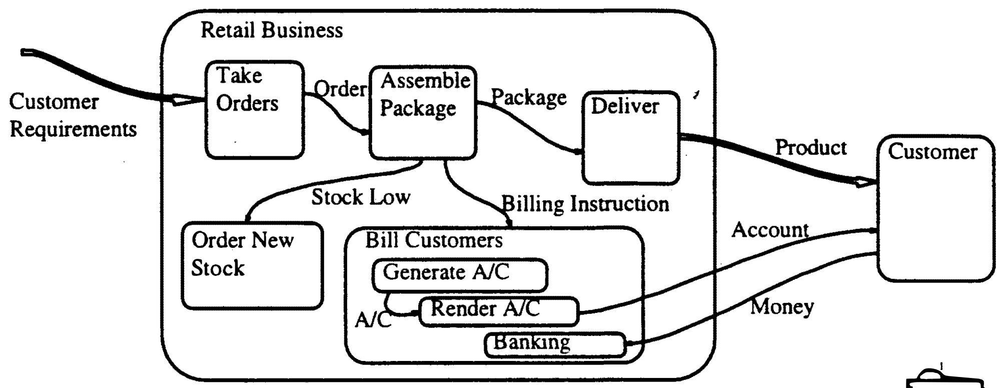
- **In common with graph and set theory, the range of applications is huge: Harel's original suggestions**
  - Entity-Relationship Diagrams,
    - blob containment = '*is-a* subtyping'
  - Data-flow Diagrams
    - blob containment = sub-task or activity
  - State Transitions
    - blobs = sub-states
  - Security Constraints
    - blobs = groups of users, files, etc.
    - edges = security privileges

Higraphs - An Overview

Entity-Relationship diagrams, data flow diagrams and state transitions may all be found in [3], whilst the security constraint processing is dealt with in [5].

Some new areas of application *1*:

- **Business process modeling**
    - blobs = process (something that delivers something useful)
    - edges = events or information flows between processes

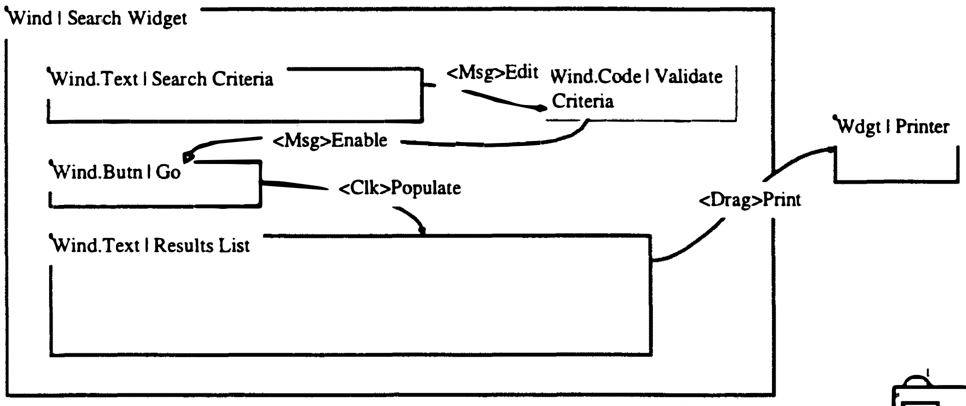


Business process modeling has been where most of our application of higraphs has been, with good reception and results. It is typically far easier to convert 'business peoples' thinking and ideas into higraphs than into standard hierarchically decomposed flow diagrams.

This example is of some of the processes of a retail business, linking the 'take order' 'assemble package' and 'deliver' processes into their 'value chain' (a term introduced by Porter [2]). The interaction with the supporting processes of 'order new stock' and Bill customer (which have a different focus to the *customer's* requirement for goods) can also be clearly shown, without confusing the 'value chain'.

Some new areas of application 2:

- **Windows Widget Message diagrams**
  - blobs = windows 'widgets' (Window, scroller, button,...)
  - edges = message or event that the widget can generate or respond to (click, drag, drop,...)
  - similar to state transition diagrams, but not quite as formal



Typically graphical user interfaces have many 'widgets' on the screen, with complex interactions between them. Specifying the interactions between them can cover many pages of text.

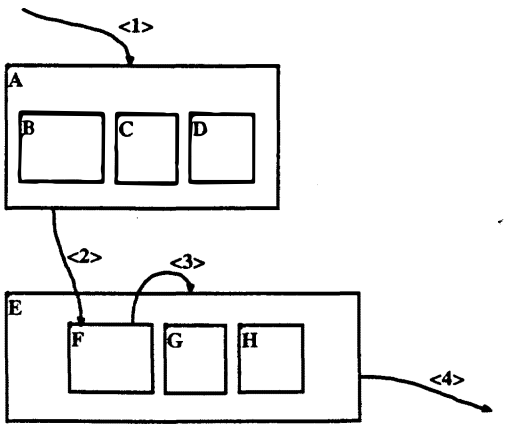
The model above allows the specification of events at a level clearly understood by programmers, and yet succinct enough to ensure correctness of design. One diagram covers flow of information and logic, draft layout of widgets, and widget containment, unambiguously.

It is also possible to break the specification down to any required level, depending on the nature of the problem and the programmer.

A complex and highly intuitive user interface has been specified and developed using this notation.

Some new areas of application *3*:

- **Specifying the domain of possible graphs, for flow optimization**



Entering A at 1,

go via B,C,D in an optimal order,

then to F,

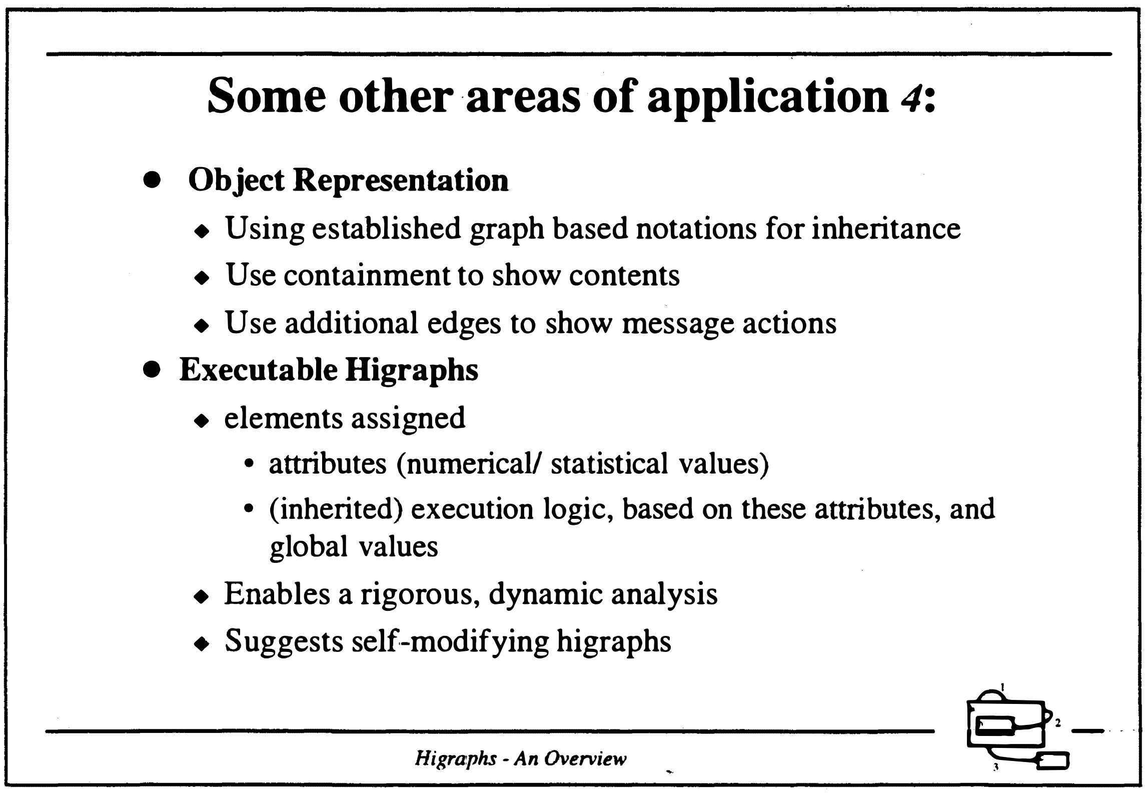
then to G & H in a optimal order,

then complete

A potential application is to use a higraph to specify the domain, or input constraints, to an optimal flow algorithm for a (weighted) walk of a graph. Typical weighting values would be time or cost. The implied semantics for the above higraph are that each blob should be visited, in some arbitrary order. The notion of concurrency between processes is also potentially implied.

The 'broadcast' type edges (*1,3*) imply that any valid successor to the start of the edge may be chosen first, but that all successors must be chosen.

This illustration is by no means complete, but is suggestive of something very useful.



Some other areas of application *4*:

- **Object Representation**
  - Using established graph based notations for inheritance
  - Use containment to show contents
  - Use additional edges to show message actions
- **Executable Higraphs**
  - elements assigned
    - attributes (numerical/ statistical values)
    - (inherited) execution logic, based on these attributes, and global values
  - Enables a rigorous, dynamic analysis
  - Suggests self-modifying higraphs

Higraphs - An Overview

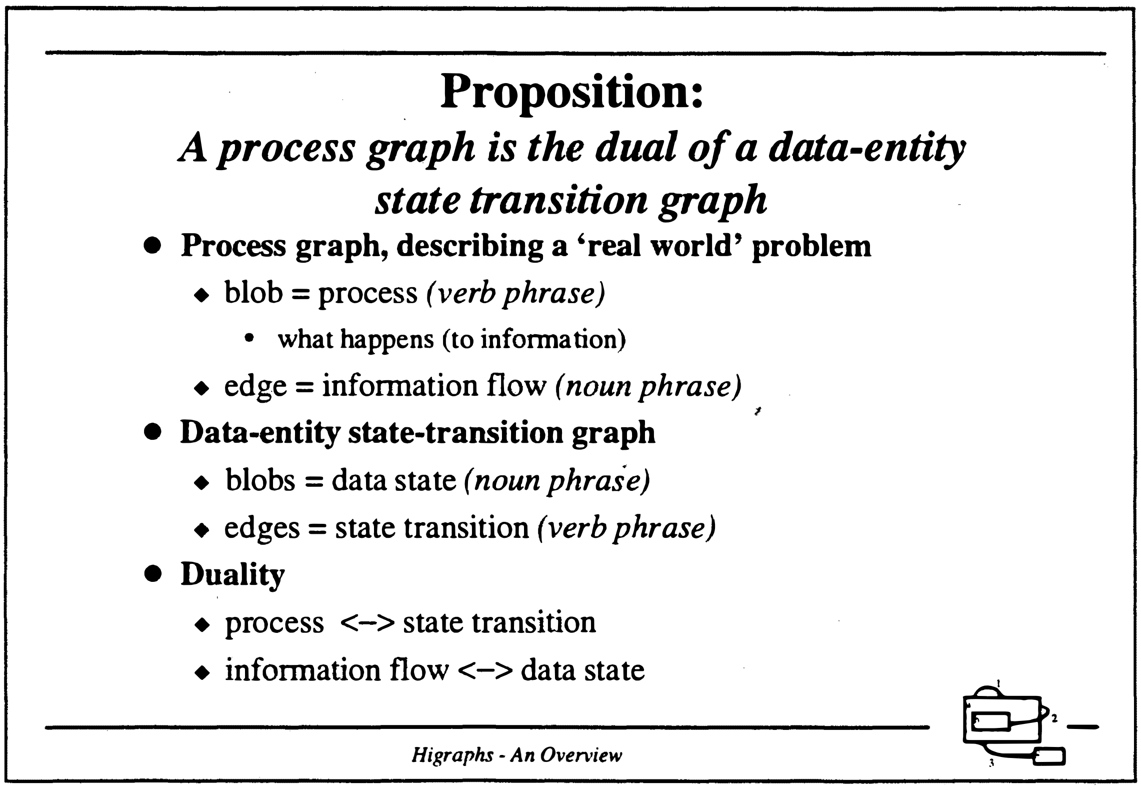
Another, totally different notation for object concepts is of dubious value, but taking an established notation, and generalizing into a higraph notation may have some benefit. Being able to clearly indicate containment is one option. This would focus more on the 'data structure' nature of the object, rather than a specifically object oriented notion.

Additional edges representing messages and function calls are also possible. Broadcast messages in particular would be elegantly and precisely handled.

In applications where concurrency is important, the orthogonality of higraphs could represent this effectively.

The notion of executable higraphs could generalize the *STATEMATE* concept of manipulating the inherent higraph elements, with some attributes. One direction could be to define an object hierarchy, with inherited execution logic for each element type in a higraph, which could then have 'instance' logic added as appropriate. The execution would proceed as the execution of the individual instances in a graph-walk order, possibly drawing on Petri-net concepts. One application of this would be simulation of processes.

Self modification in the sense of adjusting parameters, distributing values across child blobs, summarizing values into parent blobs, or even modifying the graph structure itself could be possible.



Proposition:
A process graph is the dual of a data-entity state transition graph

- **Process graph, describing a 'real world' problem**
  - blob = process *(verb phrase)*
    - what happens (to information)
  - edge = information flow *(noun phrase)*
- **Data-entity state-transition graph**
  - blobs = data state *(noun phrase)*
  - edges = state transition *(verb phrase)*
- **Duality**
  - process <–> state transition
  - information flow <–> data state

Higraphs - An Overview

The essence of this proposition was suggested out of building both types of graph for the same problem.

The process graph is more closely related to a data flow diagram at this level , but this is, in my experience, just a matter of level of detail.

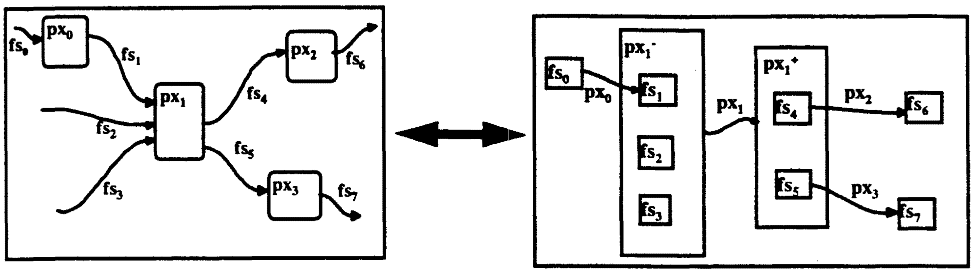
The process graph would typically be disjoint, with each connected portion of the graph having a dual for a specific set of entities, which may in turn have a non-empty intersection.

The construction and an example follow.

Proposition:

A process graph is the dual of a data-entity state transition graph

- **Construction**
  - Replace transitions with processes, flows with state of information.
  - For convergent and divergent flows, add a 'pre' (px_i^-) and 'post' (px_i^+) state.

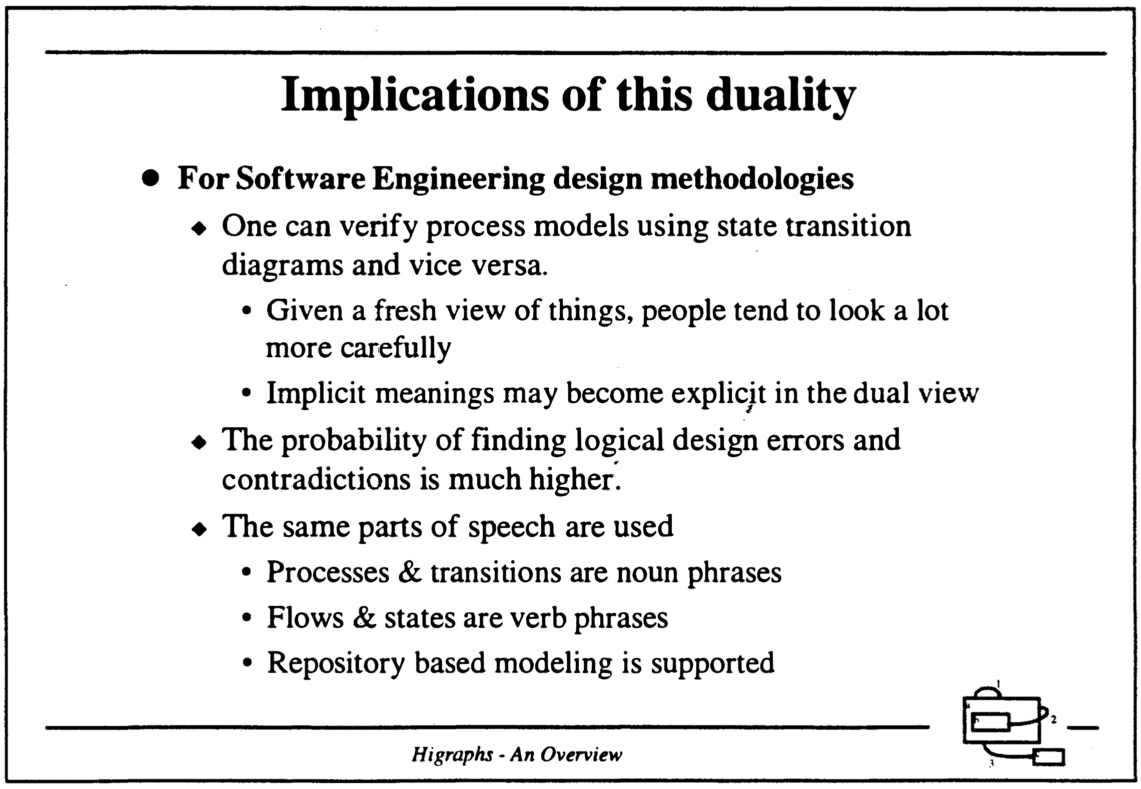


Process and transition can be considered as semantically equivalent, since they deal with the same notion of a change, in this case, to information (hence notation *px*). Similarly, flow and state can be considered equivalent (hence notation *fs*)

This proposition still requires some more analysis, but seems plausible- there may just be certain constraints on the nature of processes or entity states that are workable.

The contribution higraphs make is to allow the modeling of convergent and divergent flows. Use px_i^+ and px_i^- to represent the pre- and post- process states in a transition, encapsulating all the converging and diverging flow/ state pairs respectively.

Processes are always triggered by some sort of event- a flow or a transition, and end by delivering something- a flow or a transition. For example, a trigger could be 'Customer phones', with the end flow being 'Product delivered'. In the dual view this is consistent with there always being a start and an end state rather than transition.



Implications of this duality

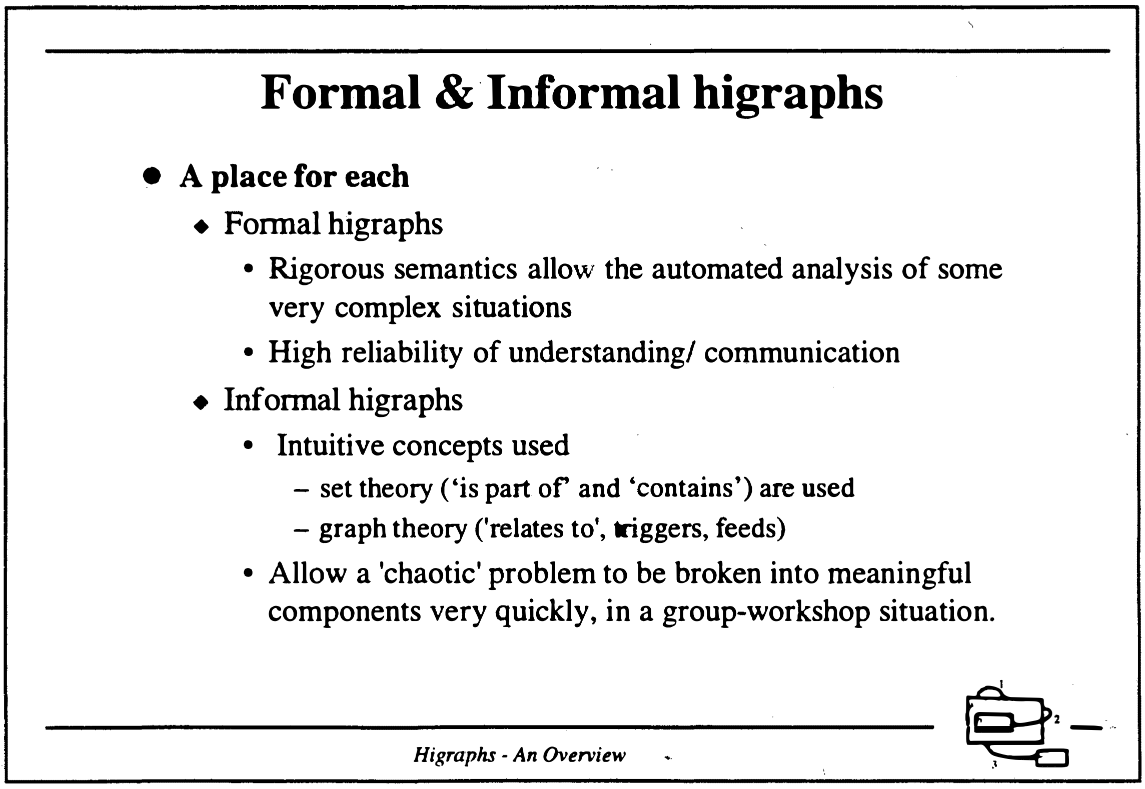
- **For Software Engineering design methodologies**
  - One can verify process models using state transition diagrams and vice versa.
    - Given a fresh view of things, people tend to look a lot more carefully
    - Implicit meanings may become explicit in the dual view
  - The probability of finding logical design errors and contradictions is much higher.
  - The same parts of speech are used
    - Processes & transitions are noun phrases
    - Flows & states are verb phrases
    - Repository based modeling is supported

Higraphs - An Overview

Software engineering, being concerned with all aspects of software development, typically considers both process (or data) flow and data state transitions. The great difficulty when working with business people who are under pressure is to get them to verify what they have done, particularly when the same information is presented in the same manner.

This duality moves the focus to the content of the information to hand, by allowing an automated translation between the dual views. The probability of finding 'design' type errors is much higher, because of this fresh presentation of the information.

Basing such work on a repository is normally almost obligatory, which would first ensure that the correct parts of speech are used, and secondly give a uniform source for all the required information.



Formal & Informal higraphs

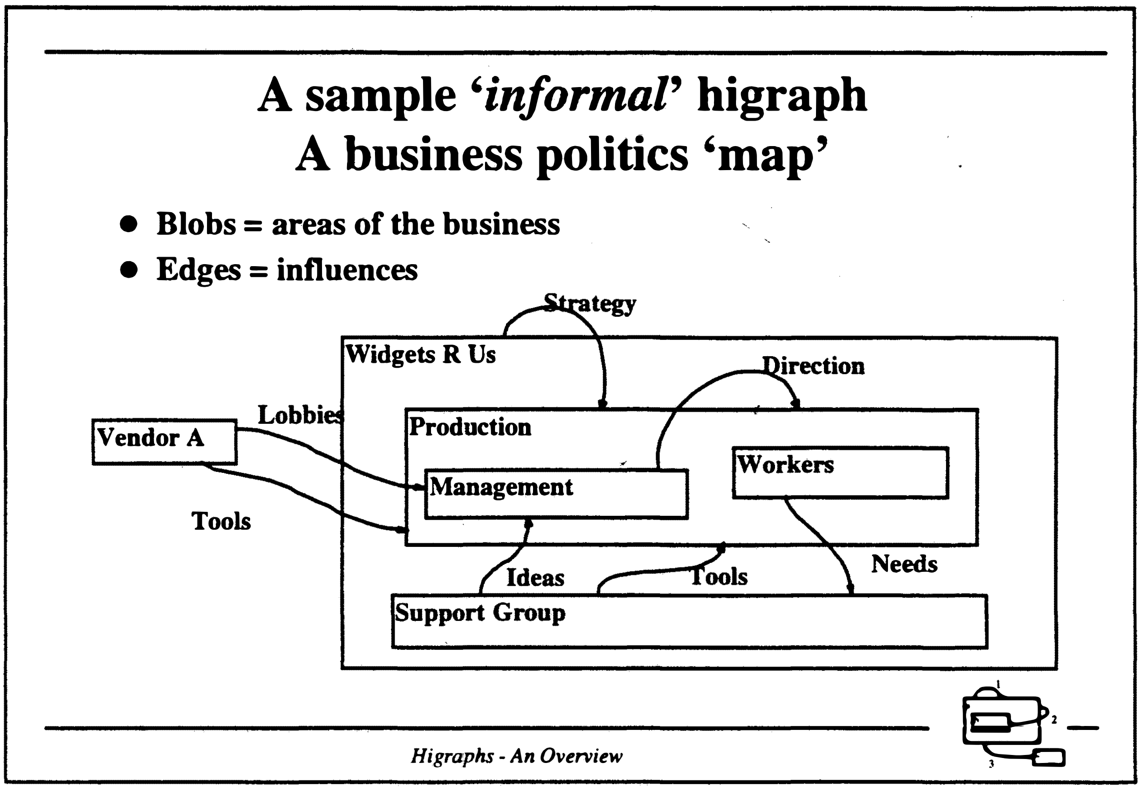
- **A place for each**
  - Formal higraphs
    - Rigorous semantics allow the automated analysis of some very complex situations
    - High reliability of understanding/ communication
  - Informal higraphs
    - Intuitive concepts used
      - set theory ('is part of' and 'contains') are used
      - graph theory ('relates to', triggers, feeds)
    - Allow a 'chaotic' problem to be broken into meaningful components very quickly, in a group-workshop situation.

Higraphs - An Overview

Higraphs with formally defined semantics have a clear place in modeling techniques. The examples outlined above are all fundamentally formal. If they have not been formally defined, it should be relatively straight-forward to do so.

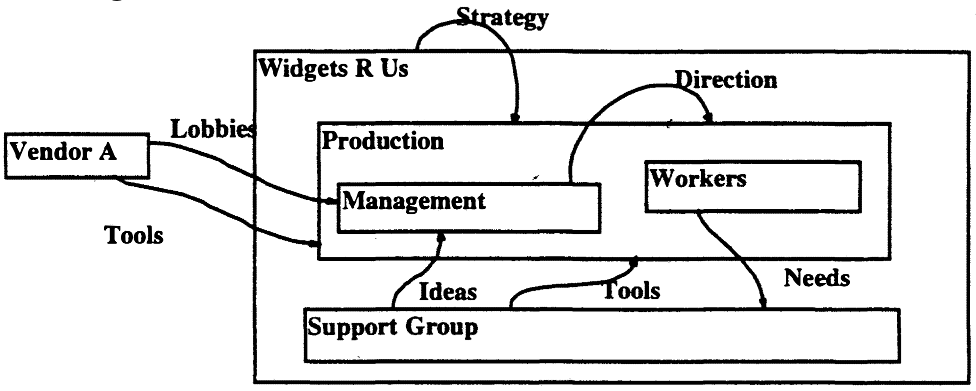
Less obvious, but as useful, is the place of higraphs with purely intuitive meanings. The set and graph theoretic foundations of higraphs are very intuitive notions, and lend themselves to developing an 'appropriate' 'semantic' 'on the fly'. 'Appropriate', 'semantic' and 'on the fly' all need to interpreted in an informal sense. But in the way that formal higraphs are intuitively understood, so a the thinking of small group can be dynamically represented using higraph notions. This can be very effective in providing a common basis for communication in such a group.

An example follows.



A sample *'informal'* higraph
A business politics 'map'

- Blobs = areas of the business
- Edges = influences

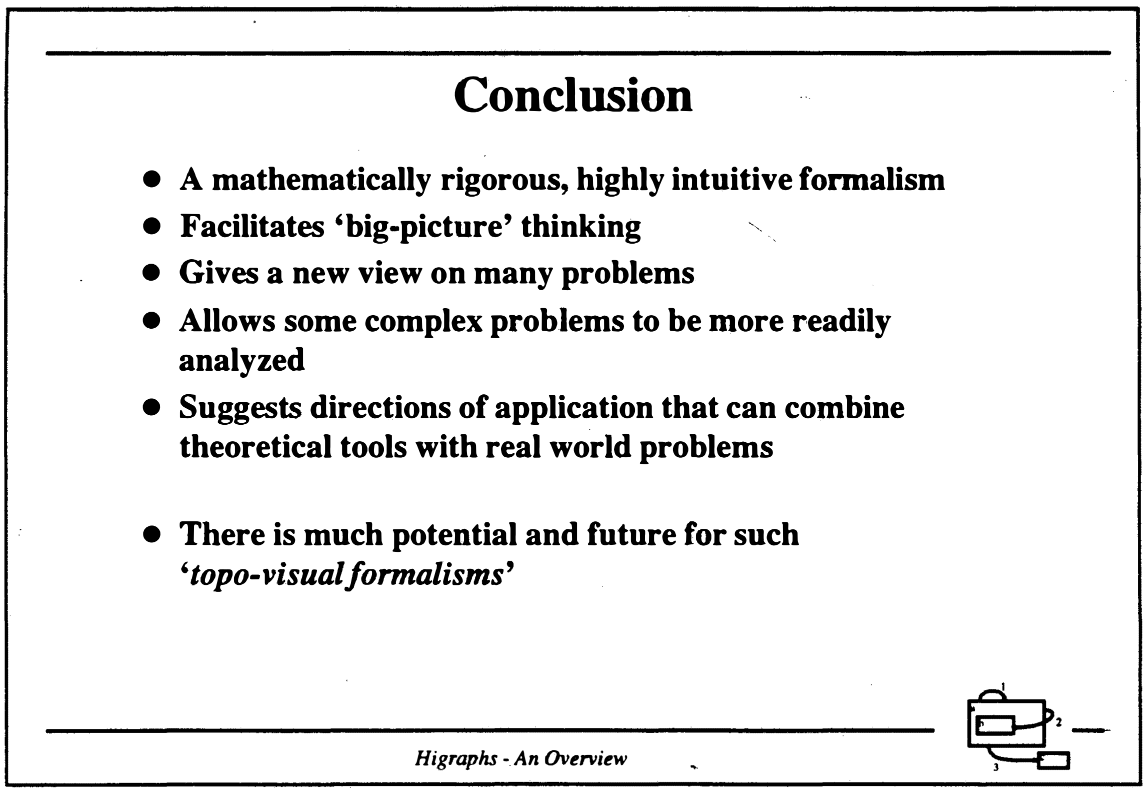


Higraphs - An Overview

This example draws on the political influences found in business.

Only that information that is immediately relevant- the tension between the vendor and the support group around tool supply- is focused on. The diagram immediately makes it clear where the problems lies, and allows the focus to be on the resolution of the tension, rather than thrashing around not understanding the problem.

As an active group facilitator, I have found this technique invaluable.



Conclusion

- A mathematically rigorous, highly intuitive formalism
- Facilitates 'big-picture' thinking
- Gives a new view on many problems
- Allows some complex problems to be more readily analyzed
- Suggests directions of application that can combine theoretical tools with real world problems
- There is much potential and future for such '*topo-visual formalisms*'

Higraphs - An Overview

Given their rigorous mathematical background, higraphs tend to be self-consistent and stable. Given their highly intuitive nature, they are easy to assimilate and understand, and so facilitate self-consistent, stable, *understood* models. This is a very valuable combination.

As the need to solve a greater number of increasingly complex problems grows, requiring a sound interaction between mind and machine, the need for techniques such as these will grow.

There is a huge potential for higraphs, both in developing formal semantics in various areas, and in applying them to a wide range of problems.

Hopefully this brief, informal overview will stimulate some further interest in the topic.

REFERENCES

Books:

[1] Tognazzini, B. 1992, *Tog on Interface*, Reading, Mass., Addison Wesley

[2] Porter, M. 1985. *Competitive Advantage*. New York: Free Press

Journal articles:

[3] Harel, D. 1988, *On Visual Formalisms*, Communications of the ACM, 31(5):514-530, May 1988

[4] Harel, D *et al*, 1990, *STATEMATE: A Working Environment for the Development of Complex Reactive Systems*, IEEE Transactions on Software Engineering, 16(4), April 1990

[5] Heydon, C.A, 1992, *Processing Visual Specifications of File System Security*, Doctoral Thesis CMU-CS-91-201

[6] Lewis, L. *Statenets, An Alternative Modeling Mechanism for Performance Analysis*, South African Computer Journal, 7:87-94, July 1992