Innovative Design as Systematic Search

Dorothy Neville & Daniel S. Weld*
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
neville, weld @cs.washington.edu

Abstract

We present a new algorithm, SIE, for designing lumped parameter models from first principles. Like the IBIS system of Williams [1989, 1990], SIE uses a qualitative representation of parameter interactions to guide its search and speed the test for working designs. But SIE's *interaction set* representation is considerably simpler than IBIS's space of potential and existing interactions. Furthermore, SIE is both complete and systematic — it explores the space of possible designs in an nonredundant manner.

Introduction

A long standing concern of Artificial Intelligence has been the automation of synthesis tasks such as planning [Allen *et al.*, 1990] and design. Of the many approaches to design (e.g., library design, parameterized design, etc.) innovative (or first principles) design has seemed to present the greatest combinatorial challenge. In this paper, we extend the work of Williams [1989, 1990] on the IBIS innovative design system. Like IBIS we assume the lumped parameter model of components and connections that is common in system dynamics [Shearer *et al.*, 1971].

We take the problem of innovative design to be the following:

- **Input:**
  1. A set of possible components (described in terms of terminals, variables, and equations relating the variables).
  2. Constraints on the number and type of legal connections between terminals.
  3. A description of an existing, incomplete device (specified as a component-connection graph).
  4. A set of equations that denote the desired behavior of the complete design.
- **Output:** A component connection graph which subsumes the existing device and whose equations are consistent and imply the desired behavior.

In this paper, we present the Systematic Interaction Explorer (SIE), an algorithm which performs this task of design from first principles. While our algorithm is based on IBIS it has a number of advantages over that algorithm:

- SIE is complete.
- SIE is systematic — it explores the space without repetition. [McAllester and Rosenblitt, 1991].
- SIE shares IBIS's interaction-focused search, yet
- SIE is small, simple, and easy to understand.

In particular, this paper presents a way to perform interaction-based invention without the complexity of IBIS's *space of existing interactions*, *space of potential interactions*, and the complex links and mappings between spaces. As explained fully below, our interaction set representation is considerably simpler than IBIS's space of potential and existing interactions, allowing us to greatly simplify the whole design algorithm. In addition, our approach results in complete and systematic exploration of the space of possible designs; we believe these properties yield greatly increased search efficiency. Although we remain unsure of the scaling potential for both IBIS and SIE, preliminary empirical results suggest that interaction-based focusing can reduce the search space by up to 95%.

In the next section, we summarize recent work on design from first principles, concentrating on William's IBIS algorithm. Then we describe the SIE algorithm and demonstrate it on the simple punchbowl example. Following that we discuss implementation status and give preliminary empirical results. We conclude with a discussion of limitations and plans for future work.

*Thanks to Franz Amador and Tony Barrett for helpful discussions. We gratefully acknowledge Oren Etzioni's emergency faxxing service. This research was funded in part by National Science Foundation Grant IRI-8957302, Office of Naval Research Grant 90-J-1904, and a grant from the Xerox corporation.

Previous Work

While there is a vast literature on design compilation, library approaches, case-based design and other approaches with restricted aims, there has been little work on design from first principles — presumably due to the combinatorics involved. Roylance [1980] backward chains from the specification equations using abstractions of primitive components, but assumes the purpose of each device and so loses completeness. Ulrich's [1988] schematic synthesis algorithm uses heuristic modifications to generate bond graphs from a specification consisting of the parameters to be related, an abstract characterization of the derivative or integral relation between the parameters, and a specification of the lumped parameter model of the input and output.

Rather than searching though the space of possible components, Williams' [1989] IBIS system searches through abstractions of this space. Specifically, IBIS constructs two graphs: the space of existing interactions and the space of potential interactions. The former is a graph whose nodes denote the value of parameters of the existing components (e.g., the pressure at the bottom of the particular vat V_1); hyperedges in the graph signify a set of parameters that are related by an equation in a component description or by a connection law such as the generalized Kirchoff's Current Law. The space of potential interactions is similar except nodes represent classes of parameters (e.g., one class might represent parameters denoting flow through pipes) and hyperedges represent relations that could be added. The two graphs are linked with edges that connect existing parameters with their respective classes. The most elegant aspect of this data structure is the way that the finite space of potential interactions represents an unbounded number of possible additions to the existing structure, yet we argue below that this very feature is also a weakness. Williams uses the interaction topology representation to aid search in three ways:

- Search control — search for interactions that are more likely to relate the parameters of the desired behavior first.
- Hierarchical testing — only consider a device worth testing when there is a path connecting all the parameters of the desired behavior.
- Verification — use information about the path connecting the parameters as a guide for verifying the desired behavior.

The key assumption made by IBIS is that a finite representation (the space of potential interactions) of the unbounded set of addable components leads to efficient search, since "Path tracing in a small graph is fast" [Williams, 1990, p. 354]. However, this ignores the effect of the resulting redundancy in search. The use of the interaction abstraction space causes IBIS to lose the property of systematic search[1] in two ways:

- There is no coordination between the debugging process of refining an inconsistent candidate and the process of generating and testing a new hyperpath from the original interaction spaces. This is crucial since "Several refinements are normally required for complex structures" [Williams, 1990, p. 355].
- No systematic way is presented for adding multiple components of a single type in service of a single objective. This can only be accomplished by repeated cycles of search and refinement [Williams, 1990, p. 354].

Since SIE searches through the concrete space of possible design topologies rather than through the abstract space of interactions, there is no need for IBIS's debugging-style refinements. This leads to a search we argue is both complete and systematic. Yet like IBIS, SIE uses the interactions of the various parameters both for search control and as a cheap method of partial design verification; thus SIE gets the same computational focus from its simple interaction set representation as does IBIS from its space of existing and potential interactions.

The SIE Design Algorithm

Our technique includes two factors that simplify the design task: interaction set representation to guide search and test potential designs, and a systematic search algorithm. We discuss the details of these below, demonstrating the technique on Williams' punchbowl example.

Defining a Device

Let $\mathcal{D} = \prec \mathcal{C}, \mathcal{N}, \mathcal{I} \succ$ be a device, where $\mathcal{C}$ is a set of *components*, $\mathcal{N}$ is a set of *nodes* (where each node is a pair[2] of component terminals signifying connections between them), and $\mathcal{I}$ is the set of *interaction sets* (explained below). The device can be *partial* if not all terminals are connected or *complete* if all terminals are connected to a node and the connection graph is connected.

For the punchbowl problem, the initial device consists of an unconnected bowl and vat: $\mathcal{C} = \{vat, bowl\}$ and $\mathcal{N} = \{\}$. The key to our algorithm is $\mathcal{I}$, a set of parameter sets; two parameters share an interaction set if and only if a change in the value of one can affect the other, i.e. if there is an interaction path (series of equations) between them. Thus the sets in $\mathcal{I}$ partition the device parameters into equivalence classes that interact causally through one or more equations. Interaction sets maintain information on which parameters can influence each

[1] Completeness may be sacrificed also, but this is unclear.

[2] The restriction to two terminals per node is relaxed in the discussion on implementation.

other without the overhead of representing the details on how they interact. Given the primitive device equations of a container:

$$\begin{aligned} V_c(t) &= H_c(t) \times area_c \\ Pd_c(t) &= fluid\_density_c \times g \times H_c(t) \\ \tfrac{d}{dt}(V_c(t)) &= Q_{top(c)}(t) + Q_{bot(c)}(t) \\ \tfrac{d}{dt}(V_c(t)) &= \tfrac{d}{dt}(H_c(t)) \times area_c \\ [area_c] &= [+] \\ [fluid\_density_c] &= [+] \\ [g] &= [+] \end{aligned}$$

SIE determines that the variables describing a container form two interaction sets. The derivatives of the fluid height and volume are related to the flow, while the fluid height is related to the pressure difference between the top and bottom of the container. The interaction sets corresponding to each unconnected component are easily generated from the primitive equations defining each component type.

Since the punchbowl initially consists of two containers, the interaction set initially consists of four parameter sets, two each for the vat and bowl.

$$\begin{aligned} \mathcal{I} = \; & \{\tfrac{d}{dt}(H_v), \tfrac{d}{dt}(V_v), Q_{top(v)}, Q_{bot(v)}\} \\ & \{Pd(vat), H_v\} \\ & \{\tfrac{d}{dt}(H_b), \tfrac{d}{dt}(V_b), Q_{top(b)}, Q_{bot(b)}\} \\ & \{Pd(bowl), H_b\} \end{aligned}$$

We use a union-find algorithm to maintain consistency of the interaction sets when joining components. When a node connects two terminals, the effort parameters (e.g. voltage or pressure) associated with the terminals get equated and the flow parameters (e.g. current) get closed with Kirchoff's Current Law (KCL). As far as the interaction sets are concerned, the only significant change has been a possible causal connection between these parameters so their respective interaction sets are unioned together.

Specifying & Testing Behavior

The desired behavior can also be considered as a set of parameters that interact in the completed device. Thus, interaction sets form a quick test of a new device's utility: do all the desirable interactions (i.e. all the parameters in the desired behavior equations) actually interact (i.e. are they all in the same interaction set)?[3]

[3]There is a potential problem with this technique. Suppose that the desired equations are $A+B=C+D$ then this could be solved by two parallel interactions $A=C$ and $B=D$ without all parameters joining a single interaction set. We can compensate for this of course with a weaker test on the interaction sets, but the focusing power is reduced. More research is necessary to formally prove necessary and sufficient interaction conditions for design validity. The IBIS algorithm has a corresponding problem — the number of hyperpaths that pairwise connect a set of parameters is vastly greater than the number of connected paths.

Algorithm: SIE$(\prec\mathcal{C},\mathcal{N},\mathcal{I}\succ, O, S, Max)$

1. **Termination:** If $|\mathcal{C}| \geq Max$ then signal failure and backtrack. Else, If O is empty and $Test(\prec\mathcal{C},\mathcal{N},\mathcal{I}\succ, S) = true$ then signal success and return design. Else, signal failure and backtrack.
2. **Select Open Terminal:** Let t be an open terminal in O.
3. **Select Connecting Terminal:** *Either* connect t to another terminal t' in O *or* instantiate new component c with terminal set O_{new} and choose t' from O_{new}. *BACKTRACK POINT: Each existing compatible open terminal and each possible new component and compatible terminal must be considered for completeness.*
4. **Update Device:** If both terminals were chosen from the existing O, let $\mathcal{C}' = \mathcal{C}$. Else, let $\mathcal{C}' = \mathcal{C} \cup \{c\}$. In either case, let $\mathcal{N}' = \mathcal{N} \cup \{(t,t')\}$.
5. **Update Interaction Sets:** If two terminals from existing components were connected, the interaction sets corresponding to the relevant parameters of the terminals are replaced with their union. If a new component was added, all of its interaction sets are added to $\mathcal{I}$, then the relevant ones are joined to reflect the connection.
6. **Update Open Terminal Set:** If both terminals were chosen from O, let $O' = O - \{t,t'\}$. Else, let $O' = O \cup O_{new} - \{t,t'\}$
7. **Recursive call:** SIE$(\prec\mathcal{C}',\mathcal{N}',\mathcal{I}'\succ, O', S, Max)$

Figure 1: The SIE Algorithm

The desired behavior for the punchbowl is "[change] the height difference in the direction opposite to the difference" [Williams, 1989, p. 59] which can be written as the following $SR1$ equation (in which square brackets denote the sign-of function):

$$[\tfrac{d}{dt}(H_v - H_b)] = [H_b - H_v]$$

This equation relates the four parameters H_v, H_b, $\frac{d}{dt}(H_v)$, and $\frac{d}{dt}(H_b)$. The first test of a potential design is to check the interaction sets of the device, ruling it out if the four parameters are not all in the same set. The quick test can definitively rule out some devices, but this is only a necessary condition. It is insufficient for complete verification. If a device passes the interaction set test, the detailed equations are generated and evaluated with respect to the desired behavior.

Generating Designs

The search algorithm takes a partial design $\prec\mathcal{C},\mathcal{N},\mathcal{I}\succ$, a list of open terminals O, and the desired behavior specification S. It systematically generates new devices by considering an open terminal and considering all the things to which it can attach: all the compatible[4] open terminals from the

[4]Representing and reasoning about compatibility is

existing components, all compatible terminals from the set of possible components and the possibility of not attaching the terminal to anything. For regularity, we consider this case as connecting the terminal to a special virtual terminal called an *endcap*, with exactly one terminal compatible with all terminal types. Figure 1 shows a non-deterministic, tail-recursive version of the algorithm. In practice, depth-first iterative deepening search can be used to implement nondeterminism.

SIE can generate Williams' solution for the punchbowl problem with four recursive calls, given the initial structure and desired behavior described previously and the open terminal set $O = \{top(vat), bot(vat), top(bowl), bot(bowl)\}$. First, SIE decides to connect terminal $bot(vat)$ to a new instance of a pipe. A pipe is defined as having a pressure difference between the ends to be proportional to the flow through the pipe. Therefore the interaction set for a pipe is one set containing the variables pressure difference an the flow at each end.

The resulting device is:

$$\begin{aligned} \mathcal{C} &= \{vat, bowl, pipe\} \\ \mathcal{N} &= \{(bot(vat), e_1(pipe))\} \\ \mathcal{I} &= \{\tfrac{d}{dt}(H_v), \tfrac{d}{dt}(V_v), Q_{top(v)}, Q_{bot(v)}, Pd_v, \\ &\quad H_v, Pd_p, Q_{e_1(p)}, Q_{e_2(p)}\} \\ &\quad \{\tfrac{d}{dt}(H_b), \tfrac{d}{dt}(V_b), Q_{top(b)}, Q_{bot(b)}\} \\ &\quad \{Pd_b, H_b\} \end{aligned}$$

The open terminal list is $O = \{top(vat), top(bowl), bot(bowl), e_2(pipe)\}$ and the second call to SIE chooses two open terminals from this set to connect, $bot(bowl)$ and $e_2(pipe)$, giving:

$$\begin{aligned} \mathcal{C} &= \{vat, bowl, pipe\} \\ \mathcal{N} &= \{(bot(vat), e_1(pipe)), (bot(bowl), \\ &\quad e_2(pipe))\} \\ \mathcal{I} &= \{H_v, V_v, Q_{top(v)}, Q_{bot(v)}, Pd_v, Pd_p, \\ &\quad Q_{e_1(p)}, Q_{e_2(p)}, H_b, V_b, Q_{top(b)}, Q_{bot(b)}, Pd_b\} \end{aligned}$$

The open terminal list is now $\{top(vat), top(bowl)\}$. The last two calls to SIE connect these in turn to a virtual endcap. With the open terminal list empty, the device is "complete" and ready to test. The interaction set test returns true for this device — all four parameters in the desired behavior are in the same interaction set. Further mathematical testing determines that indeed this device has the desired behavior.

Implementation Status & Potential

The basic SIE algorithm has been completely implemented in Common Lisp on a Sun SPARC-IPX. However, since we do not have access to an implementation of MINIMA, the final mathematical verification of potential solutions is done by hand.[5] We have tested SIE on several design problems in a domain which consists of a dozen fluid, mechanical and electrical components, including a turbine (with fluid and mechanical-rotation terminals) and a generator (with mechanical-rotation and electrical terminals). Our preliminary results are shown in figure 2.

The problems in figure 2 are summarized as follows:

- **Punchbowl.** This is the classical punchbowl example from [Williams, 1990] including the restriction that containers may not be connected directly together, and the partial device requirements that do not allow connections to be made to the tops of the existing containers.
- **Dynamo 1.** The initial device consists of an unconnected vat and a light bulb; the desired behavior relates the flow of liquid through the bottom of the vat with the light output of the bulb. SIE's solution connects the bottom of the vat to a turbine to a generator to the light bulb. The two correct solutions have the bulb's electrical terminals reversed with respect to the polarity of the generator.
- **Dynamo 2.** The same example as the dynamo, but allowing up to 5 components in the device, to illlustrate to combinatorics involved with increasing search depth. The solutions include the two previous ones and many five component solutions that have an "extra" component, such as another light bulb in series with the original one.
- **Dynamo 3.** This dynamo example has the desired behavior that the flow of fluid though the vat influences the light output of *two* lighbulbs. The correct variations have the bulbs in series with the generator.
- **Dynamo 4.** Similar to the above example with two lightbulbs, the implemention is augmented to allow for three terminals to connect to a node. Thus there are two topologically distinct solutions: the bulbs can be in series or in parallel with the generator.

Our experiments suggest that interaction sets provide the greatest performance advantage when used to evaluate devices cross technologies, with hydraulic and electrical components, for example. We predict that devices whose components are contained within one technology, will not benefit as much since all parameters will quickly collapse to the same interaction set. We plan to investigate this hypothesis with further tests.

an interesting topic in itself. Although we use a simple type system that restricts terminal connections, one could imagine a more sophisticated system such as Williams' IOTA [Williams, 1989].

[5] In the future, we intend to connect SIE to a design verification system built on top of Mathematica [Wolfram, 1988] and our PIKA simulator. [Amador *et al.*, 1993]

	DEVICES	SATISFY $\mathcal{I}$	SOLUTIONS	MAX	CPU
Punchbowl	18	3	1	3	0.167
Dynamo 1	150	4	2	4	1.550
Dynamo 2	1640	124	42	5	23.850
Dynamo 3	891	16	8	5	13.617
Dynamo 4	2786	72	14	5	39.600

Figure 2: The number of possible devices created, those that pass the interaction set test, those that pass complete mathematical verification, maximum number of components (search depth), and SPARCstation CPU time in seconds.

More Elaborate Physical Models

To evaluate this line of research, we need a clear understanding of the coverage of physical devices SIE can handle. So far we have limited ourselves to relatively simple devices with simple behavioral specifications and with one operating region. Space precludes a discussion of our algorithmic extensions to multiple behavioral regions, but see [Neville and Weld, 1992].

For simplicity, we started with the requirement that at most two terminals could connect at a node. We have extended this to allow for an arbitrary number of common connections. This increases the types of devices SIE can handle, but induces a correspondingly high combinatorical cost. Note that allowing for three-terminal nodes in the dynamo example triples the amount of time needed and the number of designs tested, while it adds only one interesting solution, the six configurations with the bulbs attached in parallel to the generator. Heuristics and search control are expected to reduce the cost. We are currently investigating the addition of search control.

Another important extension would be to incorporate geometry. While the lumped parameter model is useful and expressive for many physical processes, it fails to capture the geometric reasoning needed to design mechanical devices such as linkages and transmissions. Our design algorithm, however, is well suited for generating devices consisting of kinematic pairs or possibly unity machines [Subramanian *et al.*, 1992]. Capturing and testing geometric contraints and behavior would not be a straightforward application for interaction sets though; for analysis we would hope to draw on the ideas of Subramanian [1992] and Neville & Joskowicz [1992].

Combinatorics and SIE Scaling Potential

The most crucial question to ask of any first principles design algorithm is combinatorial: how does the approach scale? Suppose that there are C types of components, each with two terminals, and there are no restrictions on terminal connectivity except a limit of two terminals per node. Then there are $\mathcal{O}(C^n)$ connected device topologies with n symmetric parts. If more than two terminals can be connected at a node, the number of designs increases — with no limit on the number of terminals per node, then there are about $m^n C^n$ possible topologies (where m denotes the number of nodes). Considering an electric component set of identical batteries, resistors, capacitors and inductors, this suggests that there are about 17 million device topologies with 6 components and 4 nodes. While this is clearly a large number, and would take 78 hours to search with our current implementation, it is reassuring to note that existing chess machines can search this many board positions in under 10 seconds.

Note that this analysis ignores the effect of interaction representations on search. There are two ways that interaction sets increase the speed of SIE. Since the presence of all goal parameters in the same interaction set is a necessary (yet insufficient) condition for design success, interaction sets provide fast, preliminary verification technique. Of course, by itself this results in no search space reduction. The other way that interaction sets can be used is as a heuristic to guide the selection and connection of components in steps 2 and 3 of SIE (figure 1). Various heuristics are possible (maximize size of resulting interaction sets, etc.) and they correspond to search strategies in IBIS's interaction spaces.[6] The question remains: how effective are *heuristics* based on interaction sets? We believe that this question can only be answered empirically. Our hope is that the benefit will equal or surpass the speedup we have achieved in design verification.

[6] To see this, note that the combinatorial analysis of the previous section applies to IBIS as it does to SIE. For a moment assume that IBIS used a completely instantiated (infinitely large) interaction graph instead of its finite space of potential interactions. Since each component is described by one or more equations, the number of hyperedges is no less than the number of possible components. This implies that the fundamental idea of an interaction space results in no savings over search in component space — the only possible advantage comes from the use of a finite description. Yet (as we argued in section), this requires multiple refinements and the loss of systematicity. Hence we believe that IBIS searches a space that is strictly larger than SIE's space of components.

Conclusion

In this paper we have described SIE, a new algorithm for innovative design of lumped parameter models from first principles. Our approach is based on Williams IBIS system and represents an incremental advance in the search aspects of that system. We have argued that (unlike IBIS) SIE is complete and systematic. Both algorithms are sound if the subsidiary verification algorithm is sound.

We have implemented SIE and demonstrated that it runs fast enough to use it as a testbed for further research in automated design. We have demonstrated that hierarchical testing using interaction sets can eliminate up to 95 percent of the candidate devices from further expensive testing; thus it is beneficial for some types of design problems.

Our suspicion is that both IBIS's interaction spaces and SIE's interaction sets are only a partial solution to the combinatoric problems of design from first principles. We plan to continue with this research, using SIE as a testbed for search control heuristics in order to gain a better grasp of their power and the corresponding scalability of these innovative design algorithms. We suspect that in truly large design problems a first principles approach must be coupled to a library of past experience. One way to perform this is with a case-based approach that uses a modified first principles design algorithm to adapt past solutions to new problems. In [Hanks and Weld, 1992] we show how this can be done for the synthesis of partial order plans, retaining soundness, completeness and systematicity. Since we expect that it would be easy to perform the same modification on SIE, the construction of an extensive design library and a good indexing system might result in a practical design system.

References

J. Allen, J. Hendler, and A. Tate, editors. *Readings in Planning.* Morgan Kaufmann, San Mateo, CA, August 1990.

F. Amador, A. Finklestein, and D. Weld. Real-Time Self-Explanatory Simulation. *AAAI-93*, Submitted to 1993.

Steven Hanks and Daniel Weld. Systematic adaptation for case-based planning. In *Proceedings of the First International Conference on AI Planning Systems*, June 1992.

D. McAllester and D. Rosenblitt. Systematic Nonlinear Planning. In *Proceedings of AAAI-91*, pages 634–639, July 1991.

D. Neville and L. Joskowicz. A Representation Language for Conceptual Mechanism Design. In *Proceedings of the 6th International Workshop on Qualitative Reasoning*, August 1992.

D. Neville and D. Weld. Innovative Design as Systematic Search. In *Working Notes of the AAAI Fall Symposium on Design from Physical Principles*, October 1992.

G. Roylance. A Simple Model of Circuit Design. AI-TR-703, MIT AI Lab, May 1980.

J. Shearer, A. Murphy, and Richardson H. *Introduction to System Dynamics.* Addison-Wesley Publishing Company, Reading, MA, 1971.

D. Subramanian, C. Wang, S. Stoller, and A. Kapur. Conceptual Synthesis of Mechanisms from Qualitative Specifications of Behavior. In S. Kim, editor, *Creativity: Methods, Models, Tools.* 1992.

K. Ulrich. Computation and Pre-Parametric Design. AI-TR-1043, MIT AI Lab, September 1988.

B. Williams. Invention from First Principles via Topologies of Interaction. Phd thesis, MIT Artifical Intelligence Lab, June 1989.

B. Williams. Interaction-Based Invention: Designing Novel Devices from First Principles. In *Proceedings of AAAI-90*, pages 349–356, July 1990.

S. Wolfram. *Mathematica: A System for Doing Methematics by Computer.* Addison-Wesley, Redwood City, CA, 1988.