Automated Critique of Sketched Designs in Engineering

Jon Wetzel and Ken Forbus
Qualitative Reasoning Group, Northwestern University
2133 Sheridan Road, Evanston, IL, 60201, USA
jw@northwestern.edu, forbus@northwestern.edu

Abstract

Designers often use a series of sketches to explain how their design goes through different states or modes to achieve its intended function. Learning how to create such explanations turns out to be a difficult problem for engineering students. An automated "crash test dummy" to let students practice explanations would be desirable. This paper describes how to carry out a core piece of the reasoning needed in such system. We show how an open-domain sketch understanding system can be used to enter many aspects of such explanations, and how qualitative mechanics can be used to check the plausibility of the intended state transitions. The system is evaluated using a corpus of sketches based on designs from an engineering school design & communications course.

1 Introduction

One of the cornerstones of engineering education is learning to design. In the early stages of design, sketches dominate. A complex mechanism can go through multiple states or have multiple modes to achieve its intended function. To communicate how their design works, designers typically use a series of sketches, plus verbal or written information (depending on circumstance) to express information not easily sketched. According to instructors, learning how to communicate with sketches can be quite difficult for students. We are working with Northwestern's Engineering Design and Communication course (EDC) to improve students' ability to communicate using sketches. The idea is to create a *Design Buddy* for students to use in practicing explanations via sketching. The input to Design Buddy will be a sketched explanation of how their design is supposed to operate. The software's job is to scrutinize the design, and see if their explanation is plausible.

The Design Buddy is an ambitious project, and currently it is far from complete. This paper focuses on a key problem in this task: Providing feedback on explanations of intended mechanical behavior of multi-state mechanisms, entered via sketching. This problem is key because (as explained below) many designs predominantly involve forces and motion. It is a good starting point because it factors out other aspects of intent which are more open-ended (e.g., using traction pads for a device normally used in a bathroom, where surfaces are often wet) and will require additional interface modalities (e.g. text or speech) to convey.

Section 2 describes how we handle sketched input and the spatial reasoning required. Section 3 describes the qualitative mechanics reasoning involved. Section 4 describes the explanation critiquing algorithm, and Section 5 describes the evaluation on student projects[1] like the one-handed fingernail clipper in Figure 1. We close by discussing other related work and future work.

2 Sketching multi-state explanations

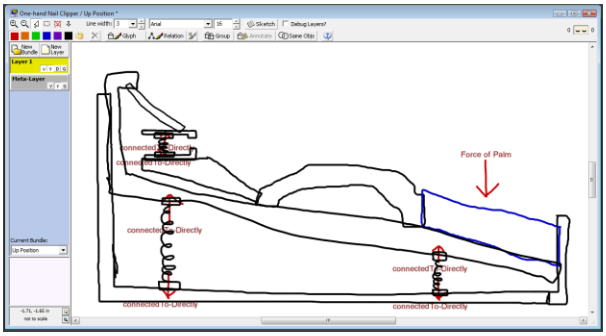


Figure 1: One-handed fingernail clipper, an EDC Project, in the up position. The hand is laid horizontally across the top, fingers pointing left, and the palm presses down to close the clipper.

We use CogSketch [Forbus *et al.*, 2008], an open-domain sketch understanding system[2], for entering and analyzing sketches. CogSketch enables users to draw *glyphs* that represent entities. A glyph is drawn by pressing a button, drawing whatever strokes constitute it, then pressing another button. This manual segmentation method is better suited for complex drawings than pen-up or time-out constraints (cf. [Cohen *et al* 1997]), because the parts of a complex

---

[1] Student projects are typically done for real customers, including patients at the Chicago Rehabilitation Institute. For instance, stroke victims often only have one working hand, which motivates several of the design tasks in the corpus.

[2] CogSketch is publicly available at http://www.silccenter.org/projects/cogsketch_index.html

design are often best drawn by multiple strokes, not always connected, and designers need to be able to take their time and think while sketching (e.g. Figure 1). What a glyph represents is indicated by labeling it with a concept from CogSketch's knowledge base (KB). This KB uses OpenCyc-derived knowledge [OpenCyC] as a starting point, so it is extremely broad (i.e., over 58,000 concepts). For example, the springs in Figure 1 are given the conceptual label **`Spring-Device`**, a concept from the KB. This is in contrast with recognition-based approaches, which require the system designer to identify in advance a small collection of entity types that can be sketched, and train recognizers for each type (cf. [Hammond & Davis, 2005]). While such systems can be useful in many circumstances, the open-ended nature of general engineering design tasks involves many more types than there are distinct visual symbols for, hence the need for another means to conceptually label them. In human to human sketching, conceptual labeling is typically accomplished via natural language. In CogSketch, a specialized interface enables users to attach KB concepts to glyphs after they are drawn. This approach means that users are never distracted by recognition errors, which tend to break their train of thought. However, it does expose them to more of the KB internals than is appropriate for a fielded system, an issue we return to in Section 7.

In addition to glyphs representing entities, CogSketch also supports *annotation glyphs* to describe an object's properties, and *relation glyphs* to describe relationships between entities. We use annotation glyphs to describe applied forces and directions of motion, using arrows. In Figure 1, for example, the force applied by the user's palm is indicated by the downward arrow on the right. Relation glyphs are used to provide a way of describing the relationships between different objects in a sketch or the different states explaining a design (see below).

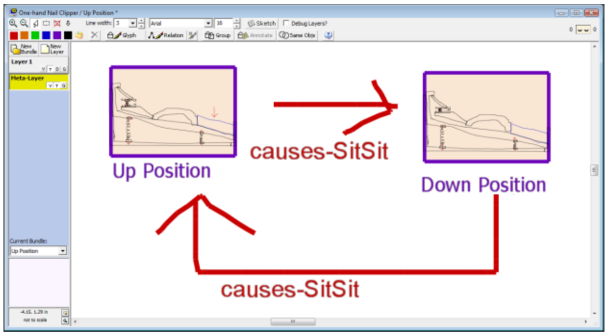


Figure 2: The metalayer provides a way to sketch multi-state explanations. Relation glyphs describe intended causal relationships between states.

CogSketch performs a variety of visual analyses on the digital ink that makes up a glyph, using techniques motivated by studies of human visual and spatial reasoning [Forbus *et al* 2008]. For example, CogSketch computes qualitative topological relationships (RCC8, [Cohn, 1996]), which we use to analyze the connectivity of parts. It also segments the ink of a glyph into lines and corners, which are used here to identify surface normals at points of contact.

In CogSketch, a sketch consists of multiple *subsketches*, each of which describes some coherent aspect of a sketch. Here subsketches are used to represent the distinct states of a design. CogSketch includes a *metalayer*, a special pane on which every subsketch of the sketch appears as an automatically-generated glyph. Multi-state explanations are entered via creating subsketches corresponding to each state, and then linking them via relationship glyphs on the metalayer. Figure 2 illustrates the explanation for the states of the one-handed fingernail clipper, the first state of which was depicted in Figure 1. The relation glyphs, each labeled with the KB relation **`causes-SitSit`** (situation causes situation), indicate that the first state will lead to the second state, and the second state will lead to a return to the first state. The second state was created by cloning the first state on the metalayer (depicted in Figure 2), then editing it by moving and resizing parts to indicate the changes therein. This can greatly simplify the sketching process, compared to pencil and paper.

3 Qualitative Mechanics

As described in [Wetzel and Forbus, 2008], we have adapted existing qualitative physics representations [Nielsen 1988][Kim 1993] for analyzing mechanisms. These representations include forces, motion, rigid objects, and the transmission of forces and movement via surface contacts. Our subsequent analysis of a corpus of student designs (see Section 5) motivated several extensions, including how forces and motion transfer across direct, rigid connections between objects, and models of springs and gears.

We use qualitative mechanics (QM) for two purposes. The first is to predict how the objects depicted in a state will behave. The second is to verify that the necessary requirements are met for each state transition to be possible. That is, given the forces that are occurring in an initial state, will the motions required to reach its proposed causal consequent actually occur?

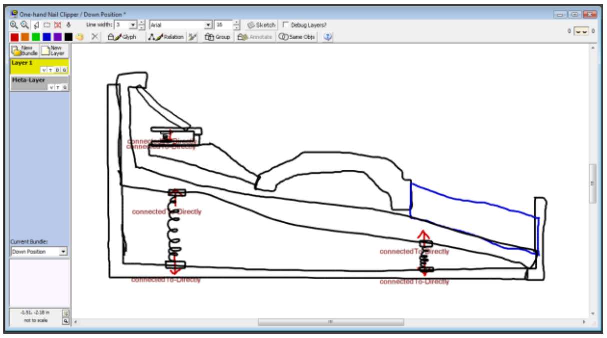


Figure 3: The "Down Position" subsketch captures the state after the clipper has been closed and the force of the palm is removed. The parts move back upward due to the compressed springs.

The connection between the entities in the sketch and QM concepts is made via conceptual labeling. For example, in Figure 1, parts which will not move relative to the sketched view are labeled with the concept **`FixedRigidObject`**. Parts which are free to move are labeled **`RigidObject`**, and the three springs are labeled as **`Spring-Device`**. The sketch also contains relation glyphs that indicate a direct connection (in the sense of glued or welded together) between objects. These relation glyphs are labeled with the relationship **`connectedTo-Directly`**. CogSketch also provides an interface for applying this relation directly to the pair of glyphs without drawing a relation glyph—we have drawn them here for illustrative purposes. An annotation glyph applied to the actuating palm rest and labeled with the concept **`forceArrow`** represents the force of the palm pressing down on the device.

As noted above, the user creates the second state (Figure 3) initially by cloning the first state on the metalayer. In the second state the palm rest is depressed, moving a latch running through the mechanism downwards that pulls the clippers closed. The springs are resized to fit the new location of the parts they are attached to, making them smaller. In order for the system to know the springs are no longer in a neutral position (currently the default) an additional conceptual label is added to the spring objects, **`CompressedSubstance`**. Finally, since the palm is no longer pressing down on the palm rest, the force annotation glyph is removed.

4. Critiquing explanations

```
CheckSketchTransitions(sketch)
For each subsketch in GetSubsketches(sketch)
 UpdateSurfaceContactKnowledge(subsketch)
For each subsketch-pair in
        GetTransitionPairs(sketch)
  For each requirement in DeduceReqs(subsketch-pair)
   For each verification in VerifyReqs(requirement)
    If verification = requirement
       then PrintSuccess(requirement, verification)
       else PrintFailure(requirement, verification)
```

Figure 4: The critique algorithm precomputes surface contact knowledge before deducing and verifying the requirements of each state transition pair (derived from the causes-SitSit relationships).

The algorithm for critiquing explanations (Figure 4) begins by using the spatial knowledge in each state to derive the set of surface contact relationships, including surface normals, between the objects in that state, using techniques from [Klenk *et al.*, 2005]. It then takes each pair of states that are linked by a causal relationship and uses an inference engine to determine what is required to transition from the antecedent state to the consequent state (**`DeduceReqs`** step, Figure 4). Currently these rules only look for motion-related differences, i.e. the appearance or lack of translation or rotation. To determine if an object has moved, the objects of type **`fixedRigidObject`** are used as reference points. For example, the glyph representing the palm rest in State 2 is lower than it was in State 1, relative to the outer frame of the device. This creates a state transition requirement that, in order for State 2 to follow from State 1, it is necessary for the palm rest to translate downwards. Similar facts are created for the other moving parts, and the same analysis is done for the transition from State 2 back to State 1.

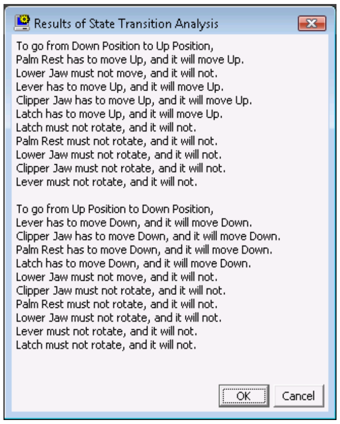
Rotations of objects between subsketches are detected in two ways. First, CogSketch automatically computes the qualitative orientation (e.g. right, up, quadrant 1, etc.) for each object in each subsketch. Looking this up is fast, but if the rotation is small the difference may not appear. If this fails, we use a cognitive model of mental rotation [Lovett *et al* 2007] to find the corresponding edges of the glyphs in each subsketch. The resulting mapping of edges is then used to calculate the angle of rotation between the glyphs. In the nail clipper example none of the parts change their orientation from state to state, so for each object the rotational requirement is that no rotation occurs.

Once the requirements for each transition have been computed, the system checks to see if they are satisfied (**`VerifyReqs`** step, Figure 4) by using qualitative mechanics to predict the next translation and rotation of the object in the antecedent state. Translation is inferred based on the constraints on the movement of the objects and the net force acting on the object. The movement constraints come from being a fixed object or being in direct contact with, or being directly connected (e.g. glue) to, another object with a constraint. The net force is found by finding all the forces acting on an object and resolving them to find the net force. The vectors used here are qualitative [Nielsen 1988], using quadrants and their edges. To help resolve ambiguities with opposing forces, the user can input a force's magnitude when creating force arrows. Both the net force and the movement constraints require the surface contact information from the sketch, which are computed at the beginning of the transition checking algorithm (Figure 4). Once they are found, if the object is free to move in a direction indicated by the net force, it will do so, otherwise it will not move. In the nail clipper sketch (Figure 2), going from State 1 (up position) to State 2 (down position), the qualitative analysis derives that the initial force will move all the free parts—from the palm rest to the upper jaw of the clipper—as drawn. For the reverse transition, the spring representation predicts that the compressed springs will provide upward forces on the other parts, causing all the parts to move upward toward their original State 1 positions. Note that the forces in State 2 did not have to be explicitly drawn as annotations by the user, as the external force in State 1 did. Instead, this force was inferred from the fact that the springs are labeled as compressed in State 2[3].

Rotation is verified in a way analogous to translation using one extra piece of knowledge: the center of rotation. Finding the center of rotation for an arbitrary object with arbitrary qualitative surface contacts and forces acting on it was beyond the current scope of this research; for now we require the user to label it with an annotation glyph. Once this is known, the torques on an object can be derived via knowing the forces on it and their relative position to the center of rotation. Similarly, rotational constraints can be derived based on surface contacts. If the object is free to rotate in a direction indicated by the net torque it will do so. Eight examples in Section 5 include instances of rotation.

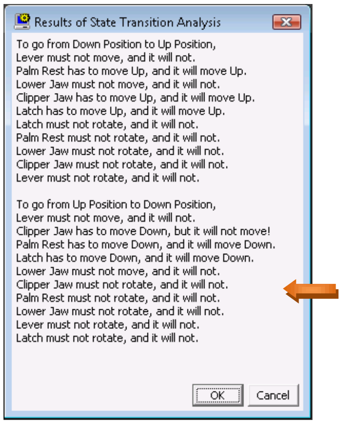
---

[3] Automatically deducing that the shorter spring in State 2 implies that it is compressed, given that the spring in State 1 is neutral, is an example of reasoning about depiction that we intend to incorporate in later versions (e.g. [Lockwood *et al* 2008]).



```
To go from Down Position to Up Position,
Palm Rest has to move Up, and it will move Up.
Lower Jaw must not move, and it will not.
Lever has to move Up, and it will move Up.
Clipper Jaw has to move Up, and it will move Up.
Latch has to move Up, and it will move Up.
Latch must not rotate, and it will not.
Palm Rest must not rotate, and it will not.
Lower Jaw must not rotate, and it will not.
Clipper Jaw must not rotate, and it will not.
Lever must not rotate, and it will not.

To go from Up Position to Down Position,
Lever has to move Down, and it will move Down.
Clipper Jaw has to move Down, and it will move Down.
Palm Rest has to move Down, and it will move Down.
Latch has to move Down, and it will move Down.
Lower Jaw must not move, and it will not.
Clipper Jaw must not rotate, and it will not.
Palm Rest must not rotate, and it will not.
Lower Jaw must not rotate, and it will not.
Lever must not rotate, and it will not.
Latch must not rotate, and it will not.
```



```
To go from Down Position to Up Position,
Lever must not move, and it will not.
Palm Rest has to move Up, and it will move Up.
Lower Jaw must not move, and it will not.
Clipper Jaw has to move Up, and it will move Up.
Latch has to move Up, and it will move Up.
Latch must not rotate, and it will not.
Palm Rest must not rotate, and it will not.
Lower Jaw must not rotate, and it will not.
Clipper Jaw must not rotate, and it will not.
Lever must not rotate, and it will not.

To go from Up Position to Down Position,
Lever must not move, and it will not.
Clipper Jaw has to move Down, but it will not move!
Palm Rest has to move Down, and it will move Down.
Latch has to move Down, and it will move Down.
Lower Jaw must not move, and it will not.
Clipper Jaw must not rotate, and it will not.
Palm Rest must not rotate, and it will not.
Lower Jaw must not rotate, and it will not.
Lever must not rotate, and it will not.
Latch must not rotate, and it will not.
```

Figure 5a(left): The explanation checks out.
Figure 5b(right): With the lever moved off to the side, the sketch violates an expectation (denoted with "!").

Finally, the system compares the results of verification with the requirements and outputs a list indicating whether they were successfully met or not. As Figure 5a illustrates, the requirements are translated into English using a simple set of templates. Figure 5b shows the output of the same system if the lever on top of the nail clipper is disconnected from the rest of the mechanism. Without it, there is nothing to exert force on the upper jaw and it will no longer move down. Violated requirements are denoted with an "!". These summaries are intended for development purposes; the NL generation for student feedback will focus on places where the system finds problems with their explanations.

5. Evaluation

The system was evaluated on examples derived from EDC projects, such as the running example of the one-handed fingernail clipper. A corpus of 39 projects was collected. 19 of these were deemed not mechanically interesting, lacking moving parts or being mainly electrical (e.g. circuits) or flow-centered (e.g. pumps). Of the 20 remaining examples, sixteen were suitable for the system. Four of them were beyond the spatial reasoning capabilities of CogSketch (mostly three-dimensional). Six of the remaining sixteen were redundant or very similar to other designs, so we performed the evaluation using only the ten designs (including the nail clipper in the earlier sections) that describe the space of problems which the system could handle.

Since the original student designs were on posters or pencil and paper, we sketched them using CogSketch ourselves. The remainder of this section highlights some of the strengths and weaknesses of the system as shown by its performance on the ten evaluation examples.

5.1 Example 1: Book Holder

Not every system in the EDC projects was intended to be a chain or sequence of states. Many projects are made to contain or stabilize something. Figure 6 shows a device designed to hold open a book. To convey this intention, we made this sketch of the desired state, cloned it and then asserted that the first state causes its copy. The system then infers that we mean for all parts in the sketch to stay stationary. The exposed page of the book has a rotational force arrow on it denoting the natural tendency for that page to flip upwards, but the clamp holds the page firmly in place. The system sees this constraint and agrees with our assertion that nothing will move.

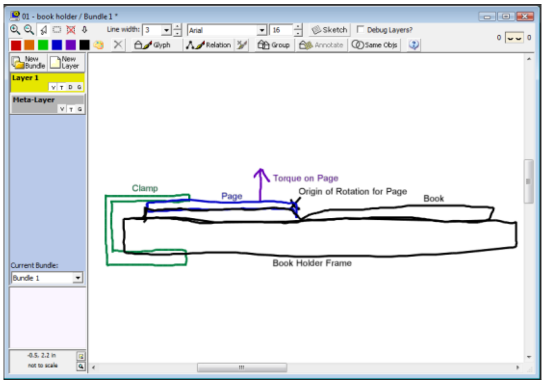


Figure 6: A book holder, viewed from the book's edge. The open page experiences an upward force, but is clamped from the left.

To test the alternate case we made another state, this time with the clamp disconnected. In this case the system warns that while the page stays stationary in our sketch, it will in reality rotate clockwise.

5.2 Example 2: Baja Mini

The Baja Mini in Figure 7 is representative of several projects that involve vehicles like go carts or solar cars. Torque on the wheels will cause it to move to the left. Without friction, the system predicts (correctly) that it will not move. When force arrows were added to represent friction, the system inferred that the whole cart could move, with the wheels pushing the frame along with them via surface contact.

Figure 7: An all-terrain vehicle in motion. Assumed torque on wheels and ground friction are required to infer motion.

5.3 Example 3: Finger Trainer

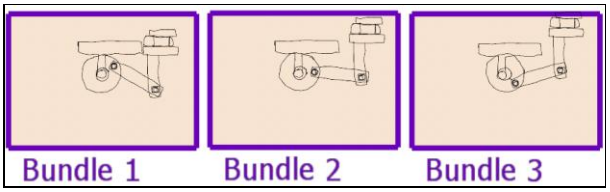


Figure 8: A device for re-training precision finger movements. The palm rests on the top with the finger stuck through a "key". The up and down movement simulates typing.

The Finger trainer (Figure 8) was difficult for the system for a couple of reasons. First, there were a number of places where parts overlapped but were not necessarily in direct contact with each other. We could draw the attachment as going around the end of the finger on the right, but in the actual project, the finger socket is a glove with the end cut off, making that drawing inaccurate. Similarly, bolts connect the different beams and the wheel but the beams and wheel have no contact with each other.

To solve this problem we will need to formally describe a three-dimensional attribute such as “inside” or “behind” to help describe the relationship between the finger and the finger slot. Also, describing the motion of the bar between the wheel and the vertical bar is difficult for the current QM because it is constrained by two different axes of rotation. Its motion will turn out to be a translation plus a rotation about some point on neither axis—we could find this point manually, but it would be laborious to require the user to do so. We plan to use Kim’s [1993] work on linkages as a basis for representing these kinds of connections between objects in the future.

5.4 Example 4: One-handed egg cracker

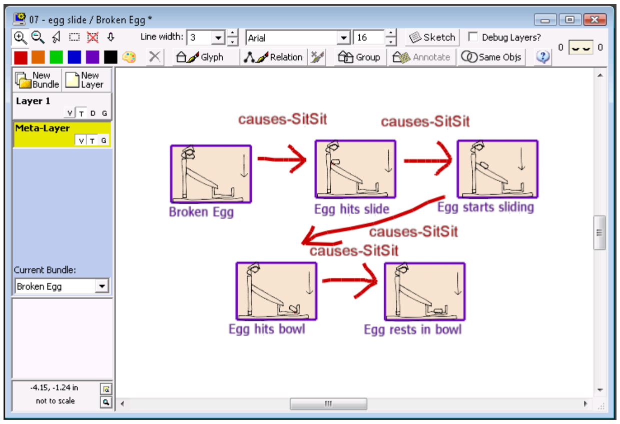


Figure 9: Device for cracking an egg with one hand. The egg shell remains in the hand (upper left) and the egg yolk slides to a bowl at the bottom of the structure.

Sketching the one-handed egg cracker (Figure 9) involved showing how the egg yolk moves down a slide and lands in a bowl at the base. While representing the process of cracking an egg is beyond the level of our QM currently, the system successfully understood the motion of the egg yolk falling, making contact with the slide, turning and sliding down the slide, making contact with the bowl, rotating and coming to rest. This example demonstrates that our system can handle a variety of translations and rotations. However, it illustrates a current weakness: it cannot reason about states which have not been drawn. There are more states here than a human partner would have required to understand the explanation, which places an extra burden on the student. We plan to investigate automatically generating new subsketches in the sketch via constrained qualitative simulation to “fill in” the implied intermediate states, to ensure that they can indeed be consistently created.

5.5 Example 5: Recliner with Shock-Absorber

To handle a non-rigid body (like the human body), the system does not try to infer what will happen to the body itself but does pays attention to any forces attributed as coming from that body. In example 5 (Figure 10) the system reasons about the behavior of the seat back, correctly predicting that it will rotate clockwise and compress the shock absorber, but it has nothing to say about the human sitting in the chair, for whom there is no QM representation yet.

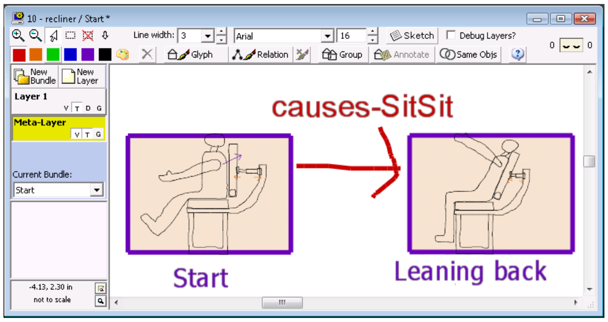


Figure 10: A reclining chair for people suffering from involuntary muscle spasms.

5.6 Example 6: Paint Roller

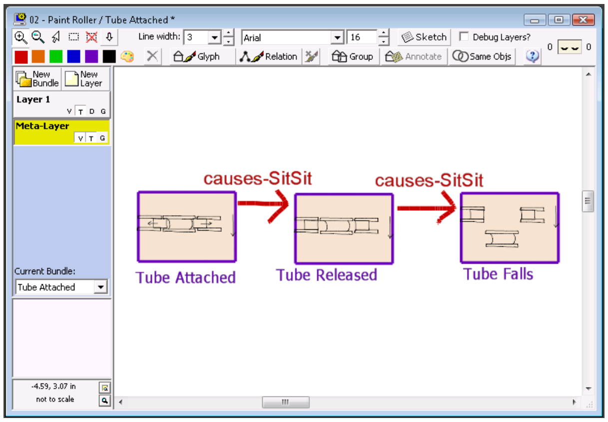


Figure 11: A quick-release paint roller. When the clamps are pulled outwards the tube falls under gravity.

Figure 11 shows a paint roller with a quick-release mechanism for changing the roll. It is drawn from a head-on perspective but might be better understood by a human if it was drawn from top down. Currently the system is limited by a lack of understanding of the conventions for illustrating depth in a drawing. If this were a top down sketch, the tube would get smaller in the third state. Work continues on interpreting these kinds of conventions.

5.7 Example 7: Ab Machine

Figure 12 shows another example of a non-rigid body at work in a sketch. This example shows a case in which our primary, qualitative method of detecting rotation is insufficient to detect a required change. The middle panel starts at about 135˚ and rotates counter-clockwise a little but not enough to be near 180˚, the next distinct qualitative direction (i.e. left rather than quadrant 2). As mentioned in Section 4, we use a model of mental rotation to confirm that this piece has actually rotated.

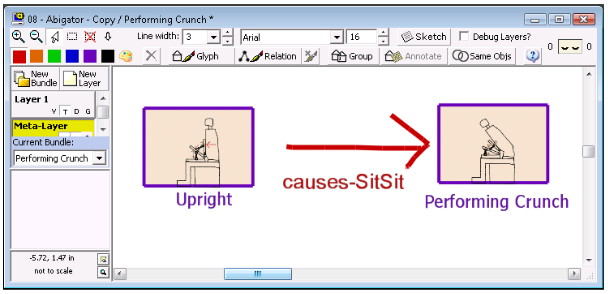


Figure 12: The device is for helping people in a wheel chair exercise their core muscles. It contains three fixed axis panels separated by springs.

5.8 Example 8: Dual-action Switch

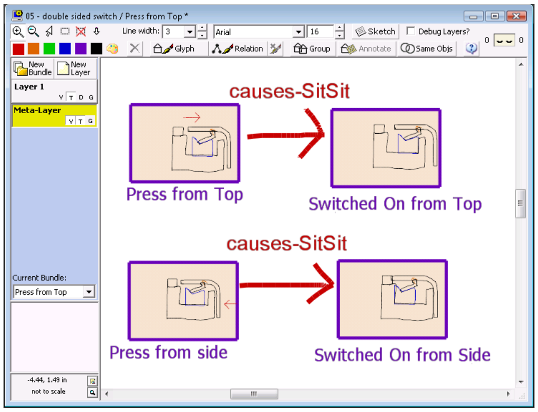


Figure 13: An electrical switch that can be activated by pressing the side or top.

The system successfully understood the mechanical aspects of this electrical switch in Figure 13. However, it is missing the greater context. There is no representation for the electrical aspects, e.g. how surface contact can transfer electric current, and the difference between a conducting surface and a non-conducting surface. The complete design sketch for this device would show these details, and while a human can infer them from looking at this sketch, it is lost on the system at this point. Many designs involve multiple domains, but we believe our state transition requirement representation is general enough to extend to those, given appropriate extensions to our knowledge base and qualitative reasoning capabilities. As we continue to work with EDC, these representations will be added and eventually be used by the `DeduceReqs` step of our algorithm (Figure 4).

5.9 Example 9: Wheelchair Softball

The wheelchair example (Figure 14) is unique in that it is drawn top-down. Students in EDC are often expected to draw their designs from side, top, and oblique perspectives. CogSketch is currently able to handle side view and top view sketches. In this case, the surface contact and force inferences worked without any extra additions to the QM knowledge. However, as discussed below, oblique perspectives are the subject of future work.

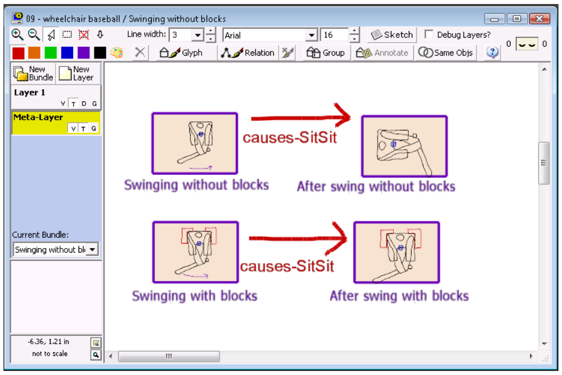


Figure 14: Rigid blocks prevent a wheelchair from rotating under the influence of swinging a baseball bat.

5.10 Example 10: Retractable Stacking Mechanism

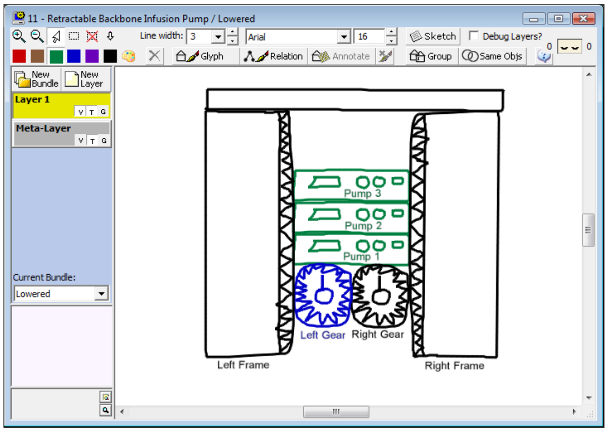


Figure 15: This retractable stacking mechanism allows pieces of medical equipment to be swapped in and out easily.

The retractable stacking mechanism in Figure 15 can be mounted on a cart for easily transporting interchangeable medical devices (in this case, backbone infusion pumps). It also demonstrates our representation of gears and toothed surfaces (drawn with zig-zag lines). When two toothed surfaces are in contact their objects are considered to be enmeshed, enabling certain behaviors. For example, when a counterclockwise torque is applied to the left gear, it rolls upwards along the fixed frame to the left. The right gear, also enmeshed with the left gear, rotates clockwise and likewise moves upward along the right frame. Together they lift up the stack of equipment until it is snug against the top of the case, preventing them from falling out.

Currently we must draw straight edges around the toothed surfaces to improve the performance of our surface-contact detection, which would otherwise have to deal with many small edges. One approach to simplifying this would be to add a perceptual model of textured edges to CogSketch, allowing it directly produce a simpler edge representation.

6. Related Work

SketchIt [Stahovich *et al* 1998] used multiple sketches linked by state transition diagrams to generate new concrete designs of fixed-axis devices, mediated by qualitative representations. Our use of sketches linked by state transitions to describe multi-state behavior is similar, but we also use them for describing alternate modes, and given the nature of our task, cannot assume that they are correct. Our qualitative mechanics reasoning is not limited to fixed-axis devices, but stays entirely at the level of sketched representations. In SketchIt users were required to identify important surface contacts, which is not unreasonable for its intended use by expert designers. Since we are dealing with novices, we must identify them automatically when possible.

Most work on sketch understanding has focused on glyph recognition, e.g., [Alvarado and Davis, 2004; Hammond & Davis, 2005; Kurtoglu and Stahovich, 2002]. Human to human sketching demonstrably does not require recognition, as anyone looking at sketches made by others without knowing the context can attest. However, recognition can act as an important catalyst, making the interaction more natural, so we would like to incorporate such techniques if further analysis indicates they could help. Recognition-based systems typically act as an interface to some traditional software system (e.g., simulation setup in [Cohen *et al* 1997] or a physics simulator [Alvarado and Davis, 2001]). Quantitative mechanical simulation would not be wise for our task, since we are focused on conceptual design, before enough information is known to support accurate numerical simulation, and inaccurate simulation would be misleading. Our use of qualitative reasoning to operate at the same conceptual level that the student is working at enables us to provide natural feedback on their explanations.

7. Future Work

The critique system described here will provide the core reasoning capability for the Design Buddy. We briefly summarize five areas where additional research is needed: extended spatial reasoning, extended qualitative mechanics, adding factors in critiquing, intent understanding, and controlled natural language processing.

Extended visual and spatial reasoning: The current techniques for computing surface contacts and axes of rotation are incomplete. Consequently, we currently use annotations to identify axes of rotation. Automating this requires improved qualitative representations of curves. Research on 3D reasoning in CogSketch is underway [Lovett *et al* 2008], which will allow us to handle perspective sketches.

Extended qualitative mechanics. The system currently only handles rigid objects plus springs and gears. We plan to use techniques from [Kim 1993] to incorporate liquids and gasses, but new theories will be needed to handle pliable solids, strings, and elastic materials. Incorporation of defaults and using broader world knowledge in model formulation is a key step. Friction is a prime example. By default one should consider friction, but choices of specific materials can be made to reduce or enhance friction, depending on the designer's intent. Adding more knowledge about materials to the KB, and appropriate default reasoning to challenge a student's explanation, will be useful steps. Our representation of the interaction of toothed surfaces could also be generalized to explain how friction causes rolling behavior.

Adding critique factors: As noted above, the state transition analysis used in generating critiques only looks at motion. There are many other relevant differences that could be included, such as changes in connection or the introduction and removal of forces. Resource consumption across paths of states can be worth monitoring for some designs. These will be added incrementally, driven by what is needed by student design projects.

Intent understanding: The current explanation input system only allows simple descriptions of intent, i.e., whether or not something moves. For the near term, we intend to continue to focus on behavioral constraints, since those can be expressed in qualitative mechanics. For the longer term, incorporating real-world motivations requires broadening of the knowledge base (e.g., that bathrooms often have wet surfaces) and more natural language input. Even then, breadth can be somewhat controlled, since those factors are often best critiqued by the student's teammates, customers for the design, and instructors.

Controlled natural language processing: While CogSketch has the ability to accept unprocessed natural language strings as labels for concepts, it currently does not provide any facility for suggesting interpretations of them in the underlying knowledge base. For conceptual labeling, we plan on using simple phrase-level techniques for inferring appropriate concepts (e.g., "spring" is the canonical pretty name for `Spring-Device` in the KB). For intent input, we plan on using a menu-based system for constructing phrases with drag & drop of sketch items for deictic reference [Forbus *et al* 2003].

Importantly, we do not have to achieve all of the above goals to start experiments with students. As our evaluation indicates, our system can already handle 25% of the typical class designs, and our collaborating instructors are willing to work with us to focus on pedagogically interesting designs within that space. Consequently, we are next focusing on automating center of rotation detection and natural language concept labeling, which should be enough for initial "pull-out" studies with EDC students in 2009. Our hope is that the work described here is a major step towards our goal, that by a combination of techniques from AI and cognitive science, engineering students will, in the long run, be able to receive help from software anytime, anyplace, in a reasonably natural way.

Acknowledgments

We thank our EDC collaborators, Ed Colgate, Bruce Ankenman, Ann McKenna, John Anderson, and Stacy Benjamin, for their valuable conversations and access to students and their portfolios. Emmet Tomai and Matthew Klenk provided valuable help with surface contact detection. This research was supported by the Spatial Intelligence and Learning Center, NSF grant SBE0541957.

References

[Alvarado and Davis, 2001] Alvarado, C., and Davis, R. 2001. Resolving ambiguities to create a natural computer-based sketching environment. *Proceedings of IJCAI01*.

[Alvarado and Davis, 2004] Alvarado, C., Davis R. 2004. Multi-domain sketch understanding. Massachusetts Institute of Technology, Cambridge, MA.

[Cohen *et al* 1997] Cohen, P., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I. Chen, L., and Clow, J. 1997. QuickSet: Multimodal interaction for simulation setup and control. *Proceedings of the 5th Conference on Applied Natural Language.*

[Cohn 1996] Cohn, A. 1996. Calculi for qualitative spatial reasoning. In *Artificial Intelligence and Symbolic Mathematical Computation*, LNCS 1138, Eds: J. Calmet, J.A. Campbell, & J. Pfalzgraf, Springer Verlag, 124-143.

[Forbus *et al* 2003] Forbus, K., Usher, J., and Chapman, V. 2003. Sketching for military courses of action diagrams. *Proceedings of IUI'03*, January, Miami, FL.

[Forbus *et al.*, 2008] Forbus, K., Lovett, A., Lockwood, K., Wetzel, J., Matuk, C., Jee, B., and Usher, J. (2008). CogSketch. *Proceedings of AAAI 2008*.

[Hammond & Davis, 2005] Hammond, T. & Davis, R. 2005 LADDER, a sketching language for user interface developers. *Computers & Graphics* 29 (518-532).

[Kim 1993] Kim, H. (1993). Qualitative reasoning about fluids and mechanics. Ph.D. dissertation and ILS Technical Report, Northwestern University. Evanston, IL.

[Klenk *et al.*, 2005] Klenk, M., Forbus, K., Tomai, E., Kim,H., and Kyckelhahn, B. (2005). Solving Everyday Physical Reasoning Problems by Analogy using Sketches. *Proceedings of 20th National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, PA.

[Kurtoglu and Stahovich, 2002] Kurtoglu T., Stahovich T.F., 2002. Interpreting Schematic Sketches Using Physical Reasoning, In *AAAI Spring*

[Lockwood *et al* 2008] Lockwood, K., Lovett, A., Forbus, K., Dehghani, M., and Usher, J. (2008). A Theory of Depiction for Sketches of Physical Systems. *Proceedings of QR 2008*.

[Lovett *et al* 2007] Lovett, A., Forbus, K., and Usher, J. 2007. Analogy with qualitative spatial representations can simulate solving Ravens' Progressive Matrices. *Proceedings of CogSci07*. Nashville, TN.

[Lovett *et al* 2008] Lovett, A., Dehghani, M., and Forbus, K. 2008. Building and comparing qualitative descriptions of three-dimensional design sketches. *Proceedings of QR 2008*. Boulder, CO.

[Nielsen 1988] Nielsen, P.E. (1988). A qualitative approach to rigid body mechanics. (Tech. Rep. No. UIUCDCS-R-88-1469; UILU-ENG-88-1775). Urbana, Illinois: University of Illinois at Urbana-Champaign, Department of Computer Science.

[OpenCyC] CyCorp, available at http://www.opencyc.org.

[Stahovich *et al.*, 1998]Stahovich T.F., Davis R., Shrobe H. 1998. Generating multiple new designs from a sketch. In *Artificial Intelligence* 104 (1998) 211–264.

[Wetzel and Forbus, 2008] Wetzel, J., Forbus, K. (2008). Integrating Open-Domain Sketch Understanding with Qualitative Two-Dimensional Rigid-Body Mechanics. *Proceedings of the 22nd International Workshop on Qualitative Reasoning*. Boulder, CO.