

Recognition of Hand Drawn Chemical Diagrams

by

Tom Yu Ouyang

Submitted to the Department of Electrical Engineering and Computer
Science

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2007

© Massachusetts Institute of Technology 2007. All rights reserved.

Author
Department of Electrical Engineering and Computer Science
May 27, 2007

Certified by
Randall Davis
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Recognition of Hand Drawn Chemical Diagrams

by

Tom Yu Ouyang

Submitted to the Department of Electrical Engineering and Computer Science
on May 27, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Chemists often use hand-drawn structural diagrams to capture and communicate ideas about organic compounds. However, the software available today for specifying these structures to a computer relies on a traditional mouse and keyboard interface, and as a result lacks the ease of use, naturalness, and speed of drawing on paper. In response, we have developed a novel sketch-based system capable of interpreting hand-drawn organic chemistry diagrams, allowing users to draw molecules with a pen-based input device in much the same way that they would on paper. The system's ability to interpret a sketch is based on knowledge about both chemistry and chemical drawing conventions. The system employs a trainable symbol recognizer incorporating both feature-based and image-based methods to locate and identify symbols in the sketch. Analysis of the spatial context around each symbol allows the system to choose among competing interpretations and determine an initial structure for the molecule. Finally, knowledge of chemistry (in particular atomic valence) enables the system to check the validity of its interpretation and, when necessary, refine it to recover from inconsistencies. We demonstrate that the system is capable of recognizing diagrams of common organic molecules and show that using domain knowledge produces a noticeable improvement in recognition accuracy.

Thesis Supervisor: Randall Davis

Title: Professor of Computer Science and Engineering

Acknowledgments

- I would like to thank my advisor Randall Davis for giving this work direction and for his support, encouragement, and guidance.
- I would like to thank the people at Pfizer Research Technology Center for their help and collaboration on this work.
- I would like to thank all the people who volunteered to participate in the user studies. Their feedback was invaluable in the development and evaluation of this work.
- I would like to thank the DHS Graduate Fellowship Program and Pfizer for their financial support.
- Finally, I want to thank my friends and family for their encouragement, especially Aaron, Jacob, Mike, Kevin, and Sonya, who provided wonderful feedback and assistance.

Contents

1	Introduction	11
1.1	Comparison to Existing Chemistry Drawing Tools	12
1.2	Approach Overview	13
1.3	Contributions	13
1.4	Outline	14
2	Making Sense of Chemical Drawings	15
2.1	Chemical Notations	15
2.2	Variations in Hand Drawn Symbols	17
2.3	Challenges in the Chemistry Domain	18
3	Stroke Segmentation and Symbol Recognition	21
3.1	Implementation Overview	21
3.2	Stroke Segmentation	22
3.3	Stroke Parsing	24
3.4	Symbol Recognition	25
3.4.1	Microsoft Handwriting Recognizer	25
3.4.2	Sequence-based Symbol Recognizer	25
3.4.3	Image-based Symbol Recognizer	28
3.4.4	Template Selection	28
3.4.5	Rotational Invariant Templates	29
3.4.6	Geometric Features	29
3.5	Support Vector Machine Classification	30

3.5.1	Scale Invariance and Feature Normalization	31
4	Structure Interpretation	33
4.1	Spatial Connectivity	33
4.2	Hypothesis Selection	34
4.2.1	Local Constraints	34
4.3	Structure Generation	35
4.4	Domain Verification	35
4.4.1	Automatic Error Correction	36
4.4.2	Balancing Structure Consistency and Interpretation Confidence	37
5	Evaluation	39
5.1	Data Collection	39
5.2	Methodology	40
5.3	Experimental Results	40
6	Related Work	45
6.1	Recognition Systems that use Domain Knowledge	46
6.2	Recognition Systems for Chemical Diagrams	47
7	Discussion	49
7.1	Future Work	49
7.2	Contributions	49

List of Figures

1-1	Left: A chemical structure drawn on a Tablet PC and interpreted using our system. Bonds are colored blue, elements are colored turquoise with their interpretation in red, and turquoise boxes represent implicit carbons atoms from bond-bond connections. Right: The same chemical structure created using ChemDraw, a popular compound authoring tool.	12
2-1	Three examples of hand drawn chemistry diagrams: (A) Penicillin, (B) Aspirin, and (C) Sildenafil.	16
2-2	Notations used in chemical diagrams. Wedge and hash bonds show the three dimensional structure of a molecule: hash bonds angle down beneath the plane, wedge bonds angle up.	16
2-3	Examples of three nitrogen, hydrogen, and hash bond symbols that demonstrate common sources of variation in how freehand shapes are drawn . . .	17
2-4	Four examples that illustrate some of the challenges (circled) in interpreting hand-drawn chemical diagrams.	19
3-1	Overview of the system architecture.	22
3-2	An example of the stroke segmentation process: dividing a stroke into conceptually separate sub-strokes which represent individual symbols. Note that the multi-scale approach allows the system to correctly classify the circled two regions in (A) as corners and the similar region in (B) as noise. . . .	23

3-3	Examples of candidate stroke groups generated by stroke parser. The left-most group corresponds to a bond in the ground truth label while the right-most group corresponds to a nitrogen symbol. The rest of the candidates represent invalid groupings of strokes. All of these candidates will be evaluated by the symbol recognizer.	24
3-4	Examples of partial symbol alignments used to reorder the strokes in a symbol.	27
3-5	Normalizing symbol orientation: The polygon approximation of the wedge bond is outlined in (A) and the principal axis of the symbol is shown in (B). The normalized shape is shown in (C).	29
4-1	Diagram showing the distance and angle measures used to identify connections between bonds and elements.	34
5-1	Irregularities in drawing style not currently handled.	41
5-2	Three examples of sketches that the system interpreted correctly.	43
5-3	Two examples where chemistry knowledge allowed the system to recover from initial errors in recognition.	44
7-1	An example of an organic reaction.	50

Chapter 1

Introduction

Visual diagrams are a means of capturing and communicating information in a variety of scientific domains, and are an important aspect of the design and problem solving process. Chemists, in particular, regularly use molecular diagrams to convey information about chemical compounds. These diagrams express a molecule's three dimensional spatial structure more directly than textual formulas, which by themselves do not readily communicate the visual properties of a compound. Structure diagrams are, as a result, often considerably more informative than the formula alone, conveying meaning about a molecule's chemical properties and potential interactions.

When chemists describe the structure of a compound to a colleague, they typically do so by sketching the diagram on paper or on a whiteboard. However, when they need to convey the same structure to a computer, they must re-create the diagram using programs that rely on traditional point-click-and-drag styles of interaction. With the growing popularity of pen based interfaces such as Tablet PCs and digital Smartboards, it is now possible to capture these types of diagrams in digital ink.

This thesis presents our work on an intelligent sketch understanding system that provides a more natural way to specify chemical structures to a computer. To preserve the familiar experience of drawing on paper, the interface supports the same set of standard chemical notations and drawing conventions. However, unlike real pen and paper, sketches created and interpreted using our system can be readily converted to structured formats that are compatible with other computer programs. This allows

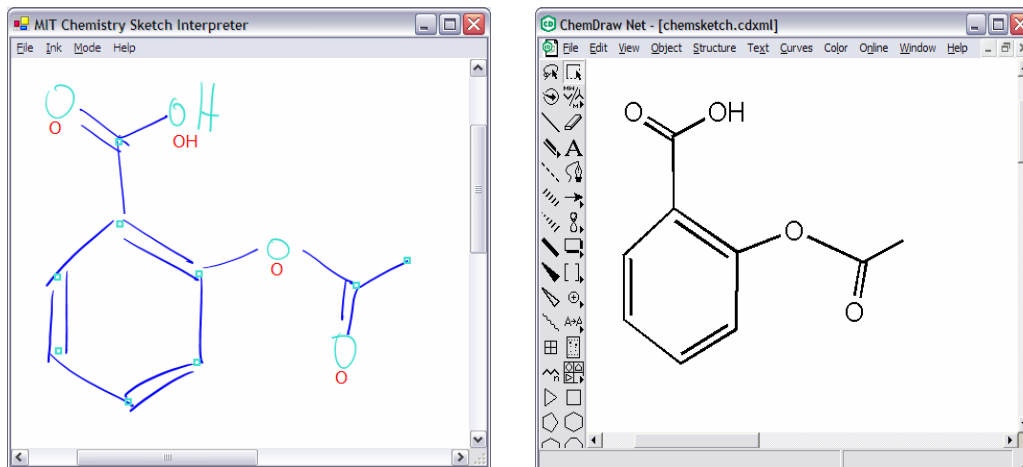


Figure 1-1: Left: A chemical structure drawn on a Tablet PC and interpreted using our system. Bonds are colored blue, elements are colored turquoise with their interpretation in red, and turquoise boxes represent implicit carbons atoms from bond-bond connections. Right: The same chemical structure created using ChemDraw, a popular compound authoring tool.

these sketched diagrams to be exported to third party software tools for tasks such as structure analysis, visualization, and database search.

1.1 Comparison to Existing Chemistry Drawing Tools

Existing software systems for specifying chemical structures rely on traditional CAD-like interfaces such as the one shown in Figure 1-1. These programs typically require that users first select a chemical component (e.g., a wedge or hash bond) from a toolbar before it can be added to the diagram. Textual information (e.g., representing elements and functional groups) is specified using the keyboard, either by typing the notation directly or by pressing a designated shortcut key.

These structure drawing programs offer many benefits over sketching on paper. They allow users to easily export their diagrams to other chemical software tools, for example in order to search for all publications or patents that mention the compound in question. They can also generate neat, typewritten diagrams that are ideal for publication. However, despite these advantages, existing systems for specifying

molecular diagrams do not provide the ease of use, naturalness, and speed of simply drawing on paper. The work we present here attempts to bridge this gap between how people naturally express chemical drawings and how computers interpret them.

1.2 Approach Overview

Our approach to this problem is inspired by how people interpret sketches of chemical structures. When we examine a sketch, we use a wide range of information: we consider the low level geometry (e.g., individual strokes) and the high level structure (e.g., the relationships between objects), and use our understanding of the semantics of the domain (e.g., what constitutes a valid configuration of atoms in a molecular compound).

While there has been a great deal of work on shape recognition in hand drawn diagrams, relatively little effort has been devoted to using knowledge about the domain to better understand and interpret a sketch. To explore this last idea, we have developed a sketch interpretation system that is aware of the rules and knowledge that govern how atoms combine to form compounds. While our implementation currently focuses on hand-drawn chemical diagrams, we believe that the approach presented here is more general and could be extended to other domains as well.

1.3 Contributions

This thesis makes three significant contributions to the field of sketch understanding and intelligent user interface design.

- First, it presents a method for recognizing sketches that contain a wide range of notations, including intermixed text, drawing, and fill-in shapes. The system uses knowledge about both chemistry and chemical drawing conventions to interpret these freehand diagrams, allowing users to draw molecules with a pen-based digital input device in much the same way that they would on paper.

- Second, it demonstrates how domain knowledge, in this case of knowledge of chemistry, can guide the sketch interpretation process, improving recognition accuracy by allowing the system to recover from low level recognition errors.
- Third, it reports the evaluation of our system on real-world examples of chemical diagrams. We collected sketches from users familiar with organic chemistry using a Tablet PC, and report the accuracy and performance of our system on these examples. There are few, if any, reported results for sketch recognition systems in our domain.

1.4 Outline

In the following chapter we will more closely examine the challenges in understanding hand drawn chemical diagrams. In Chapter 3 we describe our approach to segmenting strokes into groups that represent intended symbols, using visual and geometric features to distinguish between different shapes in the domain. In Chapter 4 we look at how the system examines the spatial context around each symbol to construct a complete chemical structure, and how it uses chemistry domain knowledge to refine its interpretation and recover from recognition errors. We present the experimental evaluation of the effectiveness of our system in Chapter 5, an overview of related work in Chapter 6, and conclude with a discussion of future work in Chapter 7.

Chapter 2

Making Sense of Chemical Drawings

This chapter begins by describing the notations and drawing conventions used in chemical diagrams, such as those shown in Figure 2-1. Next, we examine some of the difficulties in interpreting freehand sketches due to signal level noise and conceptual variations in sketched symbols. Finally, we look at the challenges in making sense of chemical diagrams, such as dealing with intermixed text and drawn shapes.

2.1 Chemical Notations

Chemical diagrams convey the shape or configuration of a molecule. This configuration is specified by atoms (indicated by their element symbols) and covalent bonds between atoms. To capture the three dimensional structure, hash bonds represent bonds which angle down beneath the plane and wedge bonds represent bonds which angle up above the plane. The vocabulary of chemical symbols that the system currently supports is outlined in Figure 2-2. Unlike gesture based approaches to sketch recognition [13, 10] which impose many constraints on how symbols can be drawn, our interface is designed to allow users to draw in much the same way as they would on paper.

In addition to the notations described above, our system accommodates two other

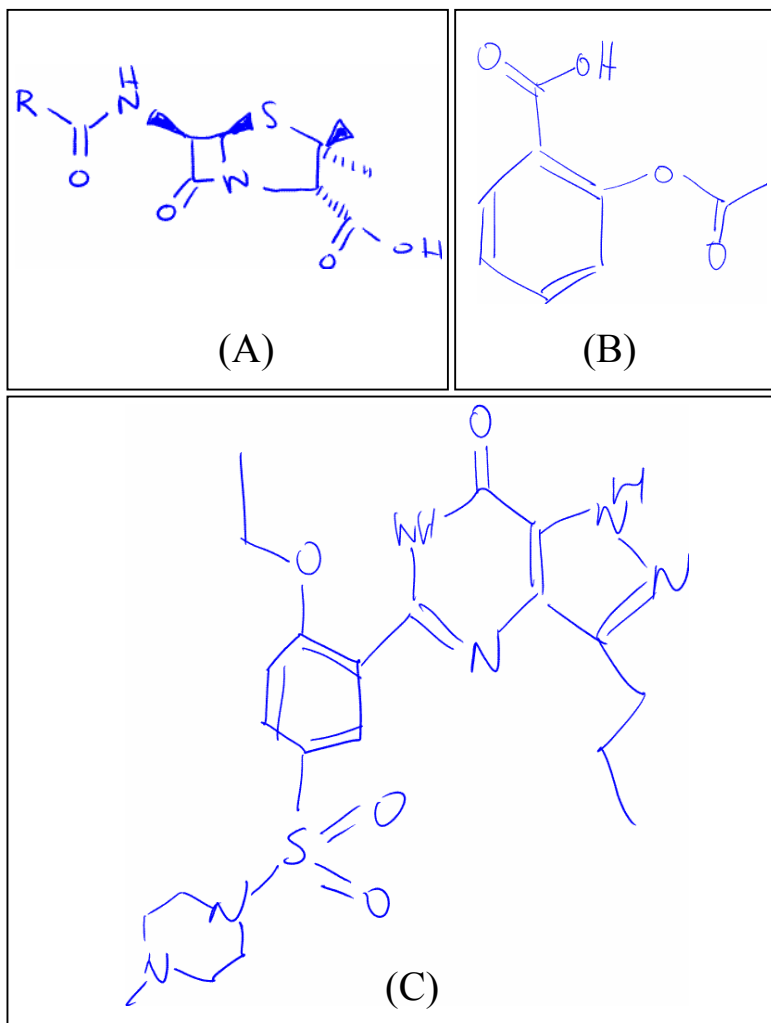


Figure 2-1: Three examples of hand drawn chemistry diagrams: (A) Penicillin, (B) Aspirin, and (C) Sildenafil.

Elements	Group Abbrev	Superscripts	Subscripts
N H	R AC	Li ⁺ O ⁻	H ₂ H ₂
Straight Bonds	Wedge Bond	Hash Bond	Aromatic Ring

Figure 2-2: Notations used in chemical diagrams. Wedge and hash bonds show the three dimensional structure of a molecule: hash bonds angle down beneath the plane, wedge bonds angle up.

common chemical drawing conventions. First, chemists frequently employ what are termed implicit elements, omitting carbon and hydrogen atoms wherever their presence can be inferred by a knowledgeable viewer. A carbon atom is implied whenever two or more bonds attached together without a connecting element, or when a bond is drawn without an attachment at one end. For each implicit carbon, enough implicit hydrogen atoms are assumed to be present to fill any vacancies in its valence shell, so that it will have the requisite four connections. In Figure 2-2, for example, the four bonds in the lower left box are all attached to one implicit carbon atom in the center.

The second special notation involves indicating an aromatic ring by drawing a circle inside a hexagonal configuration of bonds, as shown in the lower right box in Figure 2-2. These rings represent hydrocarbons with six carbon atoms connected by alternating single and double bonds. To handle this notation, the system recognizes a circle inside a 6-carbon cycle as part of an aromatic ring symbol, rather than as an oxygen atom.

2.2 Variations in Hand Drawn Symbols

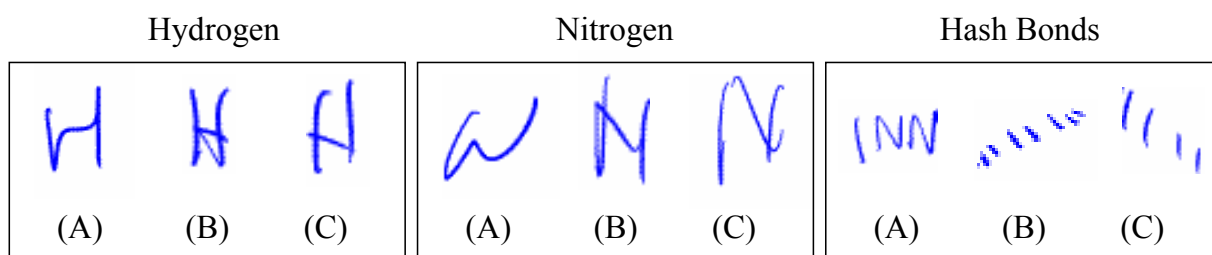


Figure 2-3: Examples of three nitrogen, hydrogen, and hash bond symbols that demonstrate common sources of variation in how freehand shapes are drawn

One of the main challenges in interpreting freehand sketches is dealing with variations in how shapes are drawn. While the task of identifying simple symbols such as straight bonds seems fairly straightforward, hand drawn sketches are often messy and imprecise. Lines that were intended to be straight may in fact be curved or jagged. Two drawings of the same symbol, even if they are from the same user, are never

perfectly uniform.

Figure 2-3 shows three examples of hydrogen, nitrogen, and hash bond symbols, each of which displays a great deal of variation compared to the other symbols of the same type. The first nitrogen symbol, for instance, is more curved than the others, and is noticeably skewed to the right. The third nitrogen, drawn with three strokes, contains a gap between the first and second stroke.

All of the symbols in each set were drawn with different numbers of strokes; for example, the first hydrogen was drawn with a single stroke, the second with two strokes, and the third with three strokes. In the middle hydrogen symbol, the user did not lift up the pen between the first and second stroke, causing an unintended ink segment connecting the vertical and horizontal segments of the *H*. We will refer to these types of differences as signal-level variations.

In addition to these kinds of unintended differences, the three hash bonds in Figure 2-3 were each drawn with different numbers of vertical dashes. These conceptual variations differ from those mentioned above since they represent multiple acceptable ways of drawing a hash bond. Signal and conceptual variations are a problem since the recognizer must be able to identify the correct class for shapes despite the differences in how they are drawn. In Chapter 3 we show how our system uses an ensemble of three symbol recognizers and general geometric features to handle these types of variations.

2.3 Challenges in the Chemistry Domain

There is a growing body of work on graphical sketch interpretation. However, there has been relatively little work that deals with the types of unconstrained diagrams found in chemical drawings. Molecular sketches similar to those shown in Figure 2-4 present a number of interesting challenges.

First, diagrams such as the one in Figure 2-4(A) contain extensively intermixed drawing and handwriting. This requires a recognition system that is not only able to determine the correct interpretation for text regions, but also distinguish between

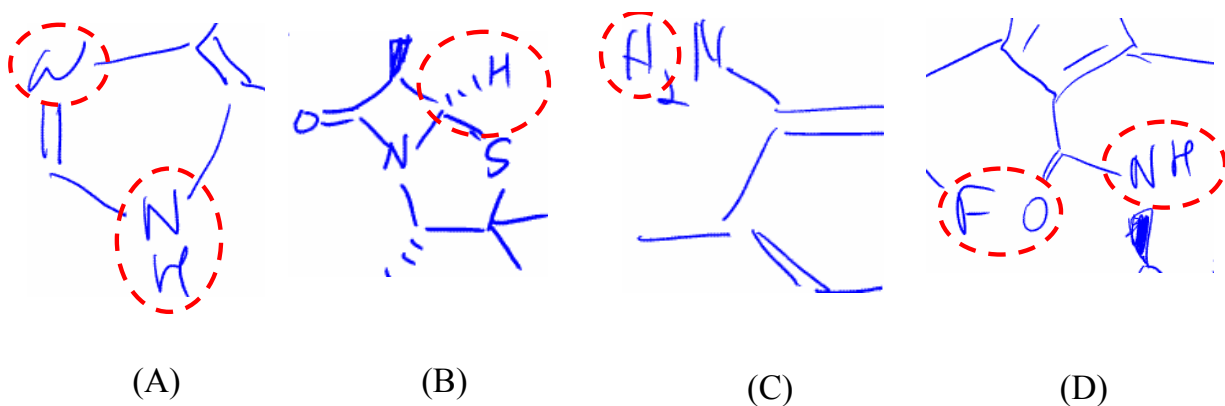


Figure 2-4: Four examples that illustrate some of the challenges (circled) in interpreting hand-drawn chemical diagrams.

text and drawn shapes.

Second, segmentation and parsing is difficult because multiple plausible interpretations may exist for different groupings of the strokes in a sketch. For example, in isolation the two vertical lines in the *H* (Hydrogen) symbol in Figure 2-4(B) might also be interpreted as either a double bond or as part of the neighboring hash bond.

Third, freehand drawings of chemical diagrams are frequently ambiguous due to inherent similarities between symbols. For example, the circled symbol in Figure 2-4(C) can reasonably be interpreted as either an *H* or an *N*. The only conceptual difference between these two symbols is the angle of the middle stroke, a distinction that may not be apparent in messy sketches such as this one.

Fourth, the structure intended by the sketch (i.e., which atoms are linked) may not be obvious. In Figure 2-4(D), the system would need to determine that the *N* and the *H* symbols should be connected while the *F* (Fluorine) and the *O* (Oxygen) should be kept separate, something not obvious from the spatial arrangement of the letters alone.

Many of these difficulties can be resolved by using a combination of spatial context (i.e., how a symbol interacts with its surrounding components) and domain knowledge (i.e., rules and constraints that specify what constitutes a valid chemical structure). For example, in Figure 2-4(C), we can use our knowledge about atomic valence to

correctly identify the circled group of strokes as a hydrogen atom, recognizing that the alternative interpretation of N_2N is not a valid chemical structure. We show in Chapter 4 how our system uses domain knowledge to resolve recognition ambiguities such as this one.

Chapter 3

Stroke Segmentation and Symbol Recognition

The previous chapter outlined the set of notations used in chemical drawings and discussed some of the challenges in identifying sketched symbols. In this chapter we begin with a broad overview of the interpretation process, followed by a description of the segmentation process and the recognition engine. Finally, we look at how the system is trained to distinguish between the different symbols in the domain.

3.1 Implementation Overview

Figure 3-1 shows the overall sketch interpretation process in our system. The first stage involves two related subproblems: ink parsing (i.e., clustering strokes into groups that represent individual symbols), and symbol recognition (i.e., determining what symbol a given group of strokes represents). Rather than perform parsing and recognition as separate steps, our system examines combinations sequentially drawn strokes up to a certain size, using the symbol recognizer to evaluate each group. In the structure interpretation stage, the system combines the resulting hypotheses about individual symbols to form a complete chemical structure, guided by domain knowledge about how chemical symbols fit together (e.g., two strokes meeting at a “T” junction cannot be bonds). Finally, during domain verification, it checks whether

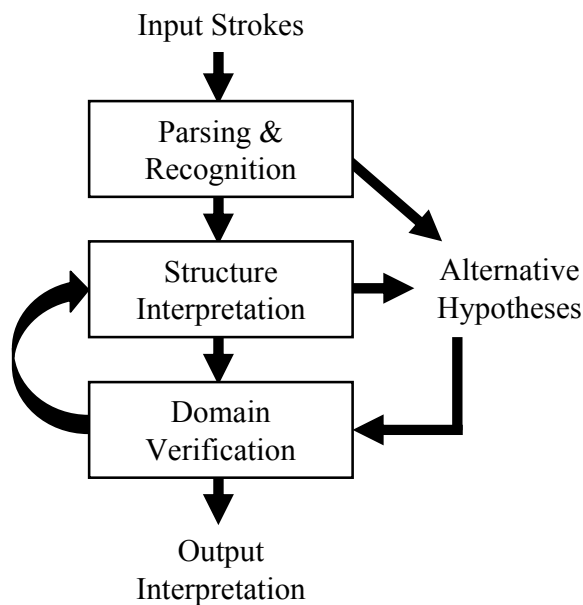


Figure 3-1: Overview of the system architecture.

this structure conforms to its knowledge of chemistry and attempts to correct any inconsistencies by reconsidering the set of alternative hypotheses.

The system carries out this process as the user draws, providing real-time feedback and updating its interpretation with each new stroke¹. Once the sketch has been interpreted, the resulting structure, expressed in a standard chemical specification format, can be passed to a variety of different programs such as ChemDraw (for rendering) or SciFinder (for querying databases of chemical properties, reactions, etc).

3.2 Stroke Segmentation

The first challenge in interpreting free-hand diagrams is stroke segmentation: dividing strokes into conceptually separate substrokes that represent independent symbols. As described in Section 2.1, chemists often draw sequences of straight bonds (connecting implicit carbons) using a single stroke. In order to accurately discern the intended structure of the diagram, the system needs to divide these multi-symbol

¹Performance is real time on a 1.5GHz Tablet PC with 1.5GB of memory

strokes into their individual components. Figure 3-2 shows two examples where the system correctly segmented a stroke into its intended subparts.

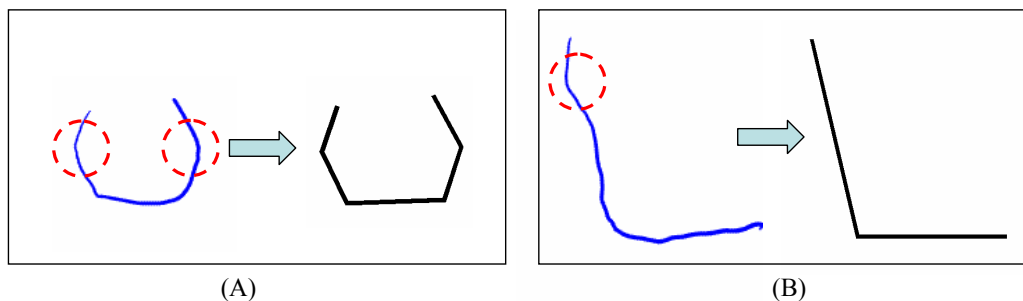


Figure 3-2: An example of the stroke segmentation process: dividing a stroke into conceptually separate substrokes which represent individual symbols. Not that the multi-scale approach allows the system to correctly classify the circled two regions in (A) as corners and the similar region in (B) as noise.

Our segmentation algorithm models such bond sequences as poly-lines, with the corner points separating the stroke into independent components. Our approach to poly-line approximation is similar to the one proposed in [14], which identifies these points by searching for regions of low pen speed and high curvature. One difference is that our method uses only curvature to identify corners. This is because we found that, for our task, including pen speed did not improve performance over using only the curvature information alone.

Another difficulty in stroke segmentation comes from the fact that the system does not know *a priori* the scale of the segments it needs to identify. If it considers any region of high curvature as a corner, it risks generating false positives, classifying variations due to signal noise as intended corners. Conversely, if the system considers only changes in orientation that persist over large regions of the stroke, it may miss corners that separate short segments. To deal with this issue we employ a multi-scale method [14] that computes the poly-line approximation at different scales, choosing the one that provides the best tradeoff between quality of fit (measured by the mean square error) and the complexity (measured by the number of predicted corners). This approach allowed the system to correctly classify the curved regions circled in Figure 3-2(A) as corners and the similar curved region in (B) as noise.

3.3 Stroke Parsing

The next task in interpreting a sketch is to parse the individual pen strokes into groups that represent valid symbols. Rather than consider all possible combinations, our system examines the strokes in the order in which they were drawn. It enumerates all possible sequence of up to n consecutively drawn strokes² and evaluates how well each group matches the symbols in the domain (described in Section 3.4 and 3.5). Figure 3-3 shows some of the candidate stroke groups generated by this process.

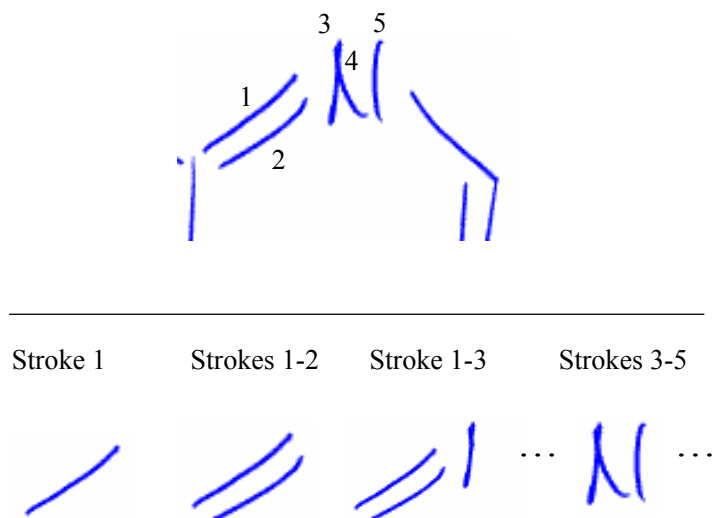


Figure 3-3: Examples of candidate stroke groups generated by stroke parser. The leftmost group corresponds to a bond in the ground truth label while the rightmost group corresponds to a nitrogen symbol. The rest of the candidates represent invalid groupings of strokes. All of these candidates will be evaluated by the symbol recognizer.

While this sliding window approach assumes that symbols are drawn with temporally contiguous strokes, we have found in our user observations that this was not a significant limitation. People rarely interspersed symbols when drawing, and in many of the cases where they did, the author was making a correction or a touch-up of a previously drawn symbol. As a result, we expect this kind of interspersing to be even less common in an interactive system where the recognizer provides real-time feedback on its interpretation.

²The current limit of $n = 7$ strokes is easily changed based on the user and the domain.

3.4 Symbol Recognition

Our approach to symbol classification is based around an ensemble of three individual symbol recognizers. In taking this multi-recognizer approach we want to allow the system to benefit from different ways of representing the information in a sketch. For example, while a time-series representation of a stroke may be suitable for distinguishing between text symbols, it may not be as appropriate for recognizing shaded shapes such as wedge bonds.

The output interpretations and confidence values from these recognizers, along with a set of geometric features, are used to train a classifier that distinguishes between the different shape classes. We will begin by describing each of these recognizers in turn, followed by an overview of the training and classification process in Section 3.5.

3.4.1 Microsoft Handwriting Recognizer

The first recognizer is the handwriting recognition system provided by the Microsoft Tablet PC SDK [6], which is used for identifying text symbols such as elements and digit subscripts. The output consists of a ranked list of recognition possibilities, each of which is assigned one of three confidence levels. Since there is information contained in both the confidence level and the rank, we combine these two measures into a single confidence score by converting the confidence level to a numerical value (*Strong* = 1.0, *Intermediate* = 0.9, *Poor* = 0.8) and dividing it by the rank. The label and confidence score of the best interpretation that corresponds to a valid chemical text symbol (i.e., element, group abbreviation, or number) is included in the classification feature vector.

3.4.2 Sequence-based Symbol Recognizer

While the Microsoft handwriting system works well for neatly printed symbols, it is less reliable on messier sketches. The confidence level and rank returned by the system provide only a coarse estimate of the match quality. It also often returns high

confidence interpretations (false positives) when evaluating non-text strokes (e.g., strokes that represent bonds or only parts of symbols). This is likely a result of its training set: it was designed for use on words and sentences, rather than individual letters and digits.

To alleviate some of the limitations of the Microsoft system, and to handle non-text chemical notations such as hash bonds, we use a second, template-based recognizer that compares the candidate input symbol or target to a set of hand drawn reference templates. This recognizer calculates the similarity between two shapes using a dynamic time warping (DTW) distance metric [5, 2, 12] based on the best possible point-to-point alignment between two sequences.

In computing the DTW distance, the target symbol (T) and reference template (R) are both normalized so that the longest dimension (width or height) of each is between $[0, 1]$. An alignment between the two shapes consists of a set of mappings $M_i = (t_i, r_i)$, where t_i and r_i represent the indices in T and R respectively. The cost of an alignment is then defined as the mean distance between each pairing of points, defined as:

$$d(t_i, r_i) = \alpha(d_\theta(t_i, r_i)/\pi)^2 + (1 - \alpha)(d_{xy}(t_i, r_i)) \quad (3.1)$$

$$d_\theta(t_i, r_i) = \text{Min}(|\theta_{t_i}^T - \theta_{r_i}^R|, |2\pi - (\theta_{t_i}^T - \theta_{r_i}^R)|) \quad (3.2)$$

$$d_{xy}(t_i, r_i) = (x_{t_i}^T - x_{r_i}^R)^2 + (y_{t_i}^T - y_{r_i}^R)^2 \quad (3.3)$$

where θ represents the orientation angle at a given point in the sequence, x and y represent the position of the point relative to the upper left corner of the bounding box of the symbol, and α is a weight³ on d_θ and d_{xy} .

We impose the restriction that the mappings in the alignment must be monotonically increasing: $r_i \geq r_{i-1}$ if $t_i \geq t_{i-1}$. Therefore, alternative alignments are generated by skipping points in either sequence if it is determined that the corresponding point in the other sequence is missing. In these cases, we add a skip penalty that also serves as an upper bound on the match distance for each pairing. The alignment with the

³ $\alpha = 0.5$ in our implementation

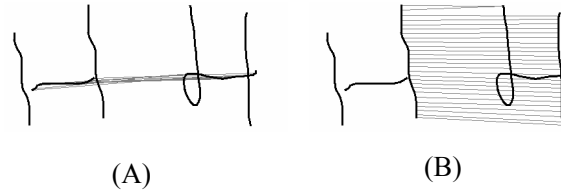


Figure 3-4: Examples of partial symbol alignments used to reorder the strokes in a symbol.

lowest distance, determined using a dynamic programming approach, is used as the match distance.

To deal with symbols drawn with more than one stroke, we take the standard approach of concatenating the individual strokes into a single sequence of points. However, one limitation of this method is its sensitivity to the manner in which symbols are drawn. For example, this algorithm can return a high match distance between two visually identical symbols if they were drawn with different stroke orders. This sensitivity also applies to the direction of the pen movement when strokes were created. According to this measure a horizontal line drawn left-to-right is different from the same line drawn right-to-left.

In order to reduce the sensitivity of the recognizer to these types of variations, our modified approach attempts to reorder the strokes in the target to better match those in the template (alternatively, if the template contains more strokes than the target, the process described below is reversed and the strokes in the template are reordered instead). It starts by computing a set of partial alignments, shown in Figure 3-4, by matching each stroke in the target in isolation. The system then reorders the strokes based on their corresponding alignment positions in the template. A stroke that matches the start of the template sequence will be placed first in the new ordering, while one that best matches the end of the sequence will be placed last. The system then performs the complete match between the reordered target and the template. To account for strokes drawn in opposite directions, the reordering process outlined above also matches the reverse sequence of each stroke against the template sequence. If the reverse sequence results in a lower match distance, it is used in the final alignment in

place of the original stroke.

The label and match distance of the best text (e.g., elements) and non-text (e.g., hash bonds) matches are included in the set of classification features. This distinction between text and non-text symbols is made to help the classifier distinguish between these two classes of notations, which we describe in more detail in Section 3.5.

3.4.3 Image-based Symbol Recognizer

Finally, we employ a third, image-based recognizer to identify shaded symbols such as wedge bonds, which are more easily characterized as an image (i.e., an area of ink) than as a sequence of points. This recognizer is based to the Tanimoto coefficient proposed in [8], which measures the degree of pixel overlap between two binary images.

Since this metric compares images rather than point sequences, the two symbols are both normalized so that their longest dimension is between [1, 24] and rendered as 24x24 binary bitmaps. The Tanimoto coefficient $T(T, R)$ is then computed as:

$$T(T, R) = \frac{n_{tr}}{n_t + n_r - n_{tr}} \quad (3.4)$$

where n_r is the total number of black pixels in the reference R , n_t is the total number of black pixels in the target T , and n_{tr} is the number of overlapping black pixels. This value measures the number of overlapping pixels between the two images normalized by the total number of black pixels in the union of the two images. The label and overlap score of the best match are included in the set of classification features.

3.4.4 Template Selection

The two template matching algorithms both require a great deal of computation, making it impractical to include every training symbol in the set of match templates. This is especially true if the recognizer is used in an interactive system, which must continually update its interpretation of the sketch as the user draws.

In order to reduce the number of templates without significantly degrading recognition accuracy, our system selects a subset of representative training examples for

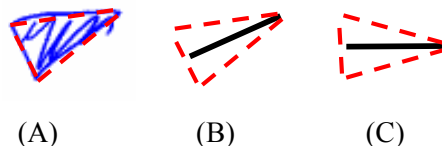


Figure 3-5: Normalizing symbol orientation: The polygon approximation of the wedge bond is outlined in (A) and the principal axis of the symbol is shown in (B). The normalized shape is shown in (C).

each of the symbols in the domain. It does this by merging similar (and therefore potentially redundant) templates using complete-link hierarchical clustering, a greedy method that iteratively combines the two clusters whose merger produces the smallest diameter. From each of these clusters, the system selects the template that is closest to the cluster center: the one with the lowest average distance to all other examples in that cluster.

3.4.5 Rotational Invariant Templates

In chemical diagrams, non-text symbols such as hash and wedge bonds can be drawn in any orientation. In order to make the template recognition process invariant to rotational differences, the system normalizes all templates for hash and wedge bond to a common orientation: with the principle axis of the symbol (representing its direction) aligned with the x-axis. We determine the principal axis of a shape by averaging the two longest edges in its polygon approximation, as shown in Figure 3-5. By normalizing the candidate group and the reference template to the same orientation before performing the match, the sequence and image based symbol recognizers are made rotationally invariant.

3.4.6 Geometric Features

We augment the confidence scores and labels produced by the three recognizers with a set of geometric features derived from the strokes in each candidate group. The purpose of these features is to help the system distinguish between similar symbols and between valid and invalid stroke groupings:

- Number of strokes: This feature takes advantage of common drawing conventions: an *O* (for oxygen), for instance, is usually drawn with a single stroke, while hash bonds typically contain at least three strokes.
- Bounding-box dimensions (a vector containing the width, height, and diagonal length of the smallest axis aligned bounding box for the candidate group): Bounding boxes of connectors (e.g., different types of bonds) are typically larger and can have a wider range of aspect ratios than bounding boxes for element symbols.
- Ink density (the ratio of the amount of ink in the candidate group to the diagonal length of its bounding box): Ink density can help indicate the type of symbol: text symbols and wedge bonds often correspond to regions of high ink density.
- Inter-stroke distance (the maximum distance between individual strokes in the group): This feature can help distinguish letters like “H” and “N” from hash bonds and double bonds.
- Poly-line approximation (average mean squared error (MSE) and average segment length of the poly-line approximations for the strokes in the group): This feature is useful for identifying straight bonds and hash bonds.
- Line-segment orientation (a vector of values that summarizes the relative orientations of line-segments in the candidate group): This feature, based on the poly-line fit, includes the number of parallel lines, perpendicular lines, and intersections between line segments. It is useful for identifying symbols made up of many line segments (e.g., hash bonds, drawn using a sequence of short parallel lines).

3.5 Support Vector Machine Classification

Once the final feature vector has been generated our recognizer trains a support vector machine (SVM) [18] to distinguish between the symbols in the domain. Our

implementation is based on a Java version of SMO [11]. Since the symbol classifier needs to distinguish between more than two classes, we use the common one-vs-one strategy for combining binary classifiers. This approach constructs a classifier for each possible pairing of the n classes. A new candidate is then classified using all $\frac{n(n+1)}{2}$ binary classifiers, and the final decision is determined by the number of votes received for each class.

We found that when training the SVM, it was useful to combine the individual textual notations (e.g., N , H , R) into an abstract *Text* class; this generalization led to a noticeable higher classification accuracy. One possible reason for this improvement is that there were insufficient examples to properly train the SVM classifier to distinguish between the full range of chemical notations. Instead, the SVM evaluates how likely a candidate group matches the general *Text* class while the exact textual labels (e.g., N , H) are determined by averaging the output from the individual recognizers in Section 3.4.

In addition to returning the most likely class label for each candidate symbol, the system also calculates a set of probability estimates for each of the classes. These probability estimates are used to rank the set of alternative interpretations for each candidate group. The process for selecting among these alternatives is described in Chapter 4.

As we discussed earlier, many of the candidate stroke groups generated by the parsing process in Section 3.3 do not correspond to intended symbols. To avoid incorrectly classifying these groups as symbols, the recognizer is also trained on invalid stroke groups that contain strokes from multiple symbols or contain only a subset of the strokes in a symbol.

3.5.1 Scale Invariance and Feature Normalization

Variations in drawing styles arise across both different users and different input devices, such as Tablet PC's and digital whiteboards (e.g., SmartBoards). As a result, the scale of the chemical diagram can vary greatly depending on the user and the sketching environment. To account for these differences, our system employs a pre-

processing step to determine the scale of the molecule by approximating the length of straight bonds in the sketch. Even though the size and shape these bonds can vary greatly, we have found that long straight line segments in the sketch typically correspond to long straight bonds in the chemical structure.

Based on this approach, the system searches for strokes with accurate poly-line approximations (low MSE) and correspond to long line segments. After removing the top 5% of these lengths as outliers, the system takes the longest remaining line as its maximum-bond-length estimate. This value is then used to normalize the scale-sensitive geometric features (e.g., bounding box dimension) listed in the previous section, making the recognizer more robust to differences in scale.

Chapter 4

Structure Interpretation

After our system has generated the set of potential candidates, it needs to choose a final set of symbols and determine the spatial structure of the resulting compound. In this chapter we describe the structure interpretation process, how the system determines relationships between neighboring components, and how these components are combined to form a complete structure.

4.1 Spatial Connectivity

The system first examines the local spatial context around each candidate symbol, evaluating three kinds of potential connections and using them to guide hypothesis selection and structure generation.

- Bond-element connection: A bond is considered to be connected to an element if the endpoint of the bond is near the element symbol and the bond points towards the element, as shown in Figure 4-1. In our implementation, the threshold on the distance d between the two components is 80% of the length of the bond, and the threshold on angle θ is 90 degrees.
- Element-element connection: An element may also combine with other elements to form compound symbols (e.g., N and H form NH). The system identifies possible element-element connections based on spatial proximity. Two elements

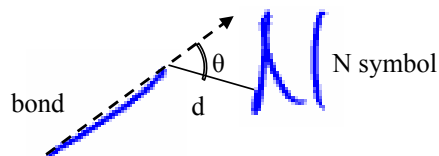


Figure 4-1: Diagram showing the distance and angle measures used to identify connections between bonds and elements.

are considered connected if the minimum distance between the two symbols is less than the diagonal length of the larger symbol.

- **Bond-bond Connection:** A bond may be connected to another bond through an implicit carbon atom. This connection is hypothesized if the distance between the endpoint of one bond is within a threshold of the endpoint of a neighboring bond, and is less than the endpoint-to-midpoint distance between the two bonds (i.e., the bonds cannot form a T junction). In our implementation, the distance threshold is 80% of the length of the longer bond.

4.2 Hypothesis Selection

The output of the parsing and recognition process often contains overlapping candidate symbols (i.e., symbols that share one or more strokes). Our system ranks these mutually exclusive candidates using the recognition certainties returned by the symbol classifier, comparing each one against all of its competing interpretations. Next, it iteratively selects candidates that offer the greatest improvement in score over their next best alternatives. The final set of predicted symbols is then passed on to the next stage of the interpretation. The system also records any alternative symbols not selected by this process, for use later in case any part of the sketch requires reinterpretation.

4.2.1 Local Constraints

While this process is running, the system also prunes candidates that are inconsistent with their local spatial context according to the following set of domain-specific

constraints:

- No dangling bonds: A candidate interpreted as a bond should be connect to at least one bond or element. The system allows the other end of the bond to be attached to an implicit carbon.
- No dangling atoms: A candidate interpreted as a chemical element should be attached to at least one bond or element.
- Subscript constraint: Numbers should appear as subscripts to an adjacent element symbol, e.g., NH_2 . A subscript should be both to the right of and below the attached symbol.
- Superscript Constraint: A charge symbol (+ or -) should appear as a superscript to an adjacent element symbol, e.g., H^+ . A superscript is both to the right of and above the attached symbol.

4.3 Structure Generation

After choosing the final set of predicted symbols, the next step is to build a connectedness graph representing the spatial structure of the compound. This process relies on the set of local relationships outlined in Section 4.1, iteratively connecting the two nearest components as long as their distance is within the appropriate threshold. Since it is relatively rare for more than three components to be mutually connected, the system uses a more conservative limit of 1/2 the original threshold when connecting components that are both already attached to other elements or bonds.

4.4 Domain Verification

The final step is to verify that the structure is chemically sound. One indication of a problem is the presence of an element with an incorrect number of bonds. For

example, a hydrogen atom can have only one bond, while a nitrogen atom has three. An inconsistency in the current interpretation can arise from three possible problems: a misinterpreted symbol (e.g., mistaking an “H” for an “N”), an error in the parsing (e.g., failing to combine strokes that belong to the same symbol), or a mistake in the connectivity (e.g., failing to connect a bond to the proper element).

4.4.1 Automatic Error Correction

Whenever an element is found with an incorrect final valence number, the system attempts to correct the inconsistency by considering each of the possible causes in turn. Since it stores alternative hypotheses from the previous stages of the interpretation process, this task involves searching the space of relevant alternatives.

If an element (including implicit carbons) is found with an incorrect final valence number, the system reconsiders the element itself (perhaps interpreting it as a different element symbol that does match the valence information) as well as any bonds or elements that are connected to it. During this process the system revives all alternative symbol candidates that meet the any of the following conditions:

- The alternative overlaps the element: selecting any of these alternatives would modify the original symbol, potentially resolving the inconsistency.
- The alternative overlaps any bond connected to the element: selecting any of these alternatives could modify the connectivity of the element, potentially resolving the inconsistency.
- The alternative is potentially connected to the element: selecting any of these alternatives could again modify the connectivity of the element, potentially resolving the inconsistency.

The refinement process starts by removing from its interpretation any symbols that overlap the set of possible alternatives described above. It then repeats the hypothesis selection process described in Section 4.2 on the affect region of the sketch.

However, instead of selecting only the best candidates and discarding any overlapping alternatives, the system searches the space of alternative interpretations as it incrementally recreates the structure.

Our system evaluates these alternatives based on the recognition cost and the chemical consistency of the resulting compound. The recognition cost is measured as the cumulative classification uncertainty for each stroke according to the SVM output. The chemical consistency cost is calculated by adding an additional penalty for each domain inconsistency discovered in the structure. If the best alternative interpretation has a lower overall cost than the original, implying that it was able to correct the inconsistency without significantly reducing the recognition quality, it is accepted and the system repeats the process for any remaining problems.

This refinement search is based on the A* algorithm, which uses a heuristic underestimate of the interpretation cost to limit the search space. Since it is difficult to determine the consistency of an incomplete chemical structure (i.e., problems discovered in the partial interpretation may be resolved by later strokes), the estimate is simply the recognition cost of the symbols selected so far best plus the recognition cost of the best interpretation for any remaining strokes. This is an underestimate of the true cost since it does not contain the added cost of chemical inconsistencies in the structure.

4.4.2 Balancing Structure Consistency and Interpretation Confidence

Based on the above description, there are cases where the system may choose not to correct a domain inconsistency if the confidence of its original interpretation is much higher than that of the chemically consistent alternative. This is done for three reasons. First, the user may still be modifying the sketch, so any inconsistencies in the structure may be only temporary. Second, the recognition process may not have generated the true candidate hypothesis, making it impossible for the system to correct the error. The system, then, should not choose a reinterpretation of the struc-

ture that is chemically consistent but unlikely to be correct according to the symbol recognizer. Third, the user may have intentionally left the structures incomplete, and hence inconsistent. It would be tiresome to use a system that always insisted on complete structures; hence we allow inconsistent structures if they are a much better description for the low level ink.

Chapter 5

Evaluation

This chapter presents the evaluation of our system on a set of chemical sketches collected from six users. It starts by describing the setup of the user study and the evaluation methodology. Next, we compare the results of our system with a simplified baseline version that does not attempt to correct inconsistencies in its interpretation. We also present examples of sketches which were interpreted correctly and discuss some of the common errors made by the system. Finally, we show examples of the system using domain knowledge to recover from previous errors in its interpretation.

5.1 Data Collection

Our goal for the evaluation was to test the performance of our system on sketches that were as unconstrained as those people produce when drawing on paper. To collect these sketches, we recruited six participants who were familiar with organic chemistry and asked each of them to draw 12 pre-specified compounds on a Tablet PC after a brief warm-up. The data collection interface was designed to behave like a piece of paper, i.e., capturing the sketch but providing no recognition or feedback. During the study, a reference diagram of the molecule was presented at the upper right corner of the tablet display. This was done to indicate the structure to be drawn without requiring that the user knew the compound from memory. To prevent participants from simply copying the structure, the reference image disappeared whenever the

user’s stylus came near the drawing surface.

We believe this provided a stringent, real-world test, inhibiting the tendency of users to draw more carefully when they know their strokes are being interpreted. It also avoids the tendency for users to adapt their drawing style to the recognizer when they see their strokes being misinterpreted.

5.2 Methodology

We performed a set of user-independent performance evaluations for our system. In these tests, we ran our system on each of the sketches, treating the examples from the other five users as training data for the symbol classifier and as the source of templates for the sequence-based and image-based recognizers. By leaving out sketches from the same user, these tests indicate how our system would perform for a new user who has not provided any training data to the system.

To calculate recognition accuracy we compared final set of symbols identified by our system with those in the hand-labeled ground truth. To be considered a true positive, a retrieved symbol needs to match both the location (stroke grouping) and the classification of the ground truth. Performance of the system is measured by the number of symbols identified correctly in each sketch.

5.3 Experimental Results

Method	Precision	Recall	F-measure
Domain	93.1%	94.1%	93.6%
Baseline	91.7%	93.1%	92.4%

Table 5.1: Overall recognition accuracy of the domain-enabled and baseline systems for the six users in the study.

We report the precision, recall, and F-measure in Table 5.1. Figure 5-2 shows three examples of sketches that the system interpreted correctly. We also compare our system’s performance against a simplified baseline version that has no knowledge

of atomic valence and hence does not attempt to correct chemical inconsistencies. The results show an overall improvement in accuracy when the system uses chemistry domain knowledge to improve its interpretation. The system reduced the number of missed symbols by 14% over the baseline. Using paired t-tests, we find that domain knowledge enabled system significantly outperforms the baseline version ($p < .05$).

Figure 5-3 shows two examples where domain knowledge allowed the system to recover from low level recognition errors. In the example on the left, the system mistakenly interpreted the N symbols as groups of 3 bonds. This misinterpretation resulted in 3 bonds being connected to an H_2 compound, which is inconsistent with the chemical rules regarding valence. The chemistry verification step noted this error and correctly reinterpreted the strokes as an N instead. In the second example on the right, the system made two mistakes: it misinterpreted the N on the left as an H and mis-parsed the N on the right, classifying the rightmost vertical stroke as a bond. The first mistake caused a chemical inconsistency where 3 bonds were connected to a hydrogen atom, while the second mistake resulted in four bonds connected to a nitrogen atom. The system again recognized and corrected these errors using its knowledge of chemistry.

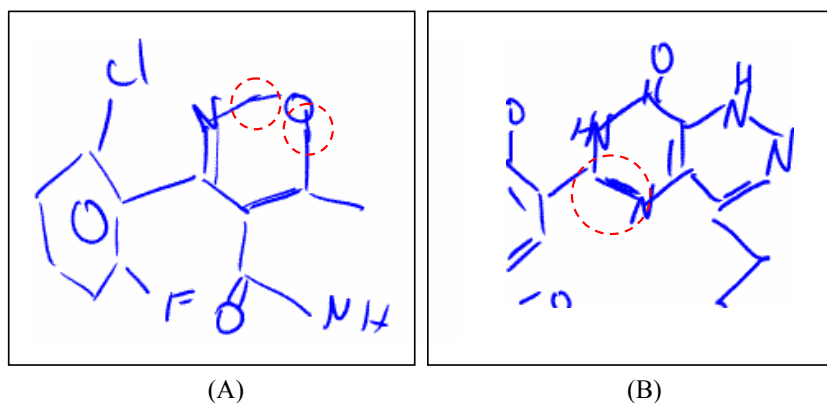


Figure 5-1: Irregularities in drawing style not currently handled.

An inspection of the output revealed that a number of recognition errors were caused by symbols that the system was unable to parse correctly. For example, in Figure 5-1(A) on two separate occasions the author drew a single bond using

two strokes (highlighted by the dashed circles). Our system currently assumes that straight bonds are drawn with at most one stroke, and as a result classified the two strokes as two individual, connected bonds. In Figure 5-1(B), the user over-traced one of the bonds (in the dashed circle), causing the system to incorrectly interpret the stroke as two connected bonds.

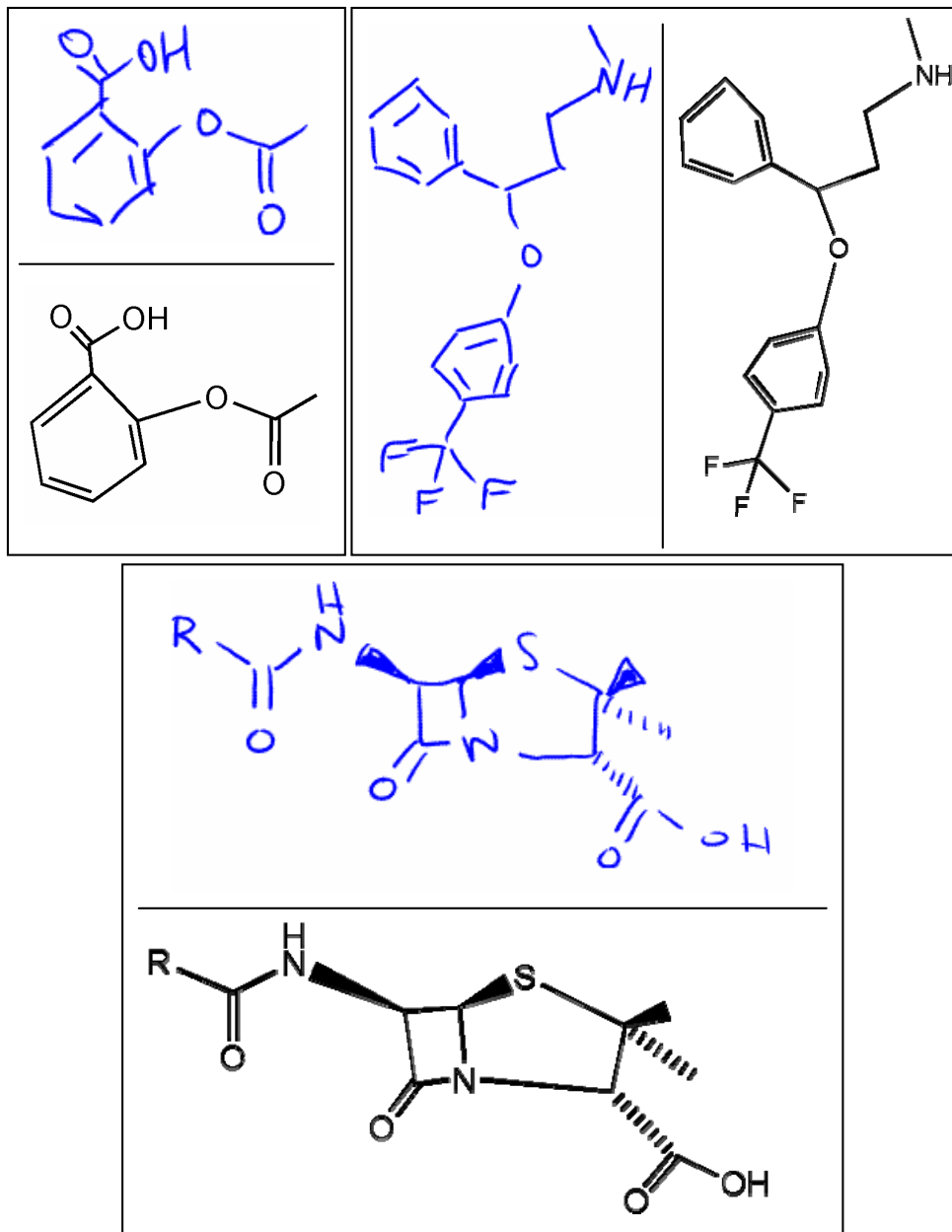


Figure 5-2: Three examples of sketches that the system interpreted correctly.

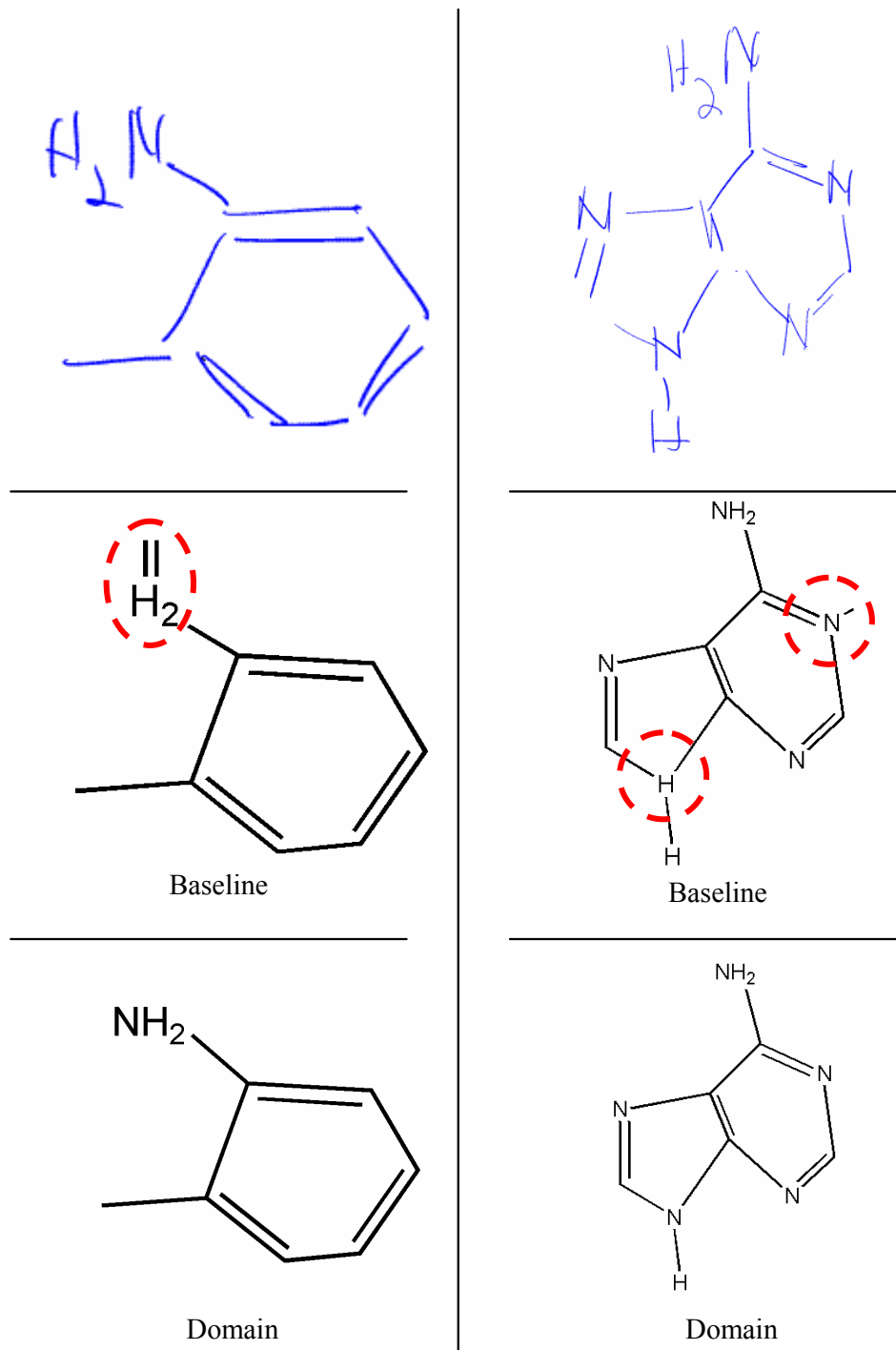


Figure 5-3: Two examples where chemistry knowledge allowed the system to recover from initial errors in recognition.

Chapter 6

Related Work

Some of the early work in sketch recognition [13] focused on recognizing single stroke gestures, or glyphs. Symbols are recognized based on simple features such as the angle of the stroke or the area of its bounding box. Long *et al.* [10] developed a system that analyzes the similarity between symbols to help developers design gestures that will not be easily confused by the computer. One limitation of gesture based systems is that they place strict restrictions on how symbols can be drawn, often using a single stroke or a pre-defined sequence of stroke.

Another approach to sketch recognition is based on hierarchical shape descriptions, with the lowest level composed of geometric primitives such as lines, ellipses, and arcs. Graph based techniques [3] model these primitives and the relationships between them as graphs, with recognition posed as a graph isomorphism problem. Shilman *et al.* [15] presented an approach that uses a hand coded visual grammar to describe shapes in the domain. Recognition is then treated as a statistical visual language parsing problem. Alvarado and Davis [1] proposed an approach that uses dynamically constructed Bayesian networks to parse a sketch, employing both top-down and bottom-up interpretation.

Shilman and Viola [16] presented a system for recognizing and grouping text and graphics in diagrams and equations. Their approach links individual strokes into a proximity graph and searches for symbols among spatially connected subgraphs. In contrast to our system, their approach was evaluated on synthetically generated

geometric shapes, flowcharts, and equations, and does not use domain knowledge to refine its interpretation.

6.1 Recognition Systems that use Domain Knowledge

Gennari, Kara, and Stahovich [7] developed a sketch based interface that uses geometry and domain knowledge to interpret hand drawn electronic circuit diagrams. They employ a set of geometric heuristics (e.g., density, bounding box size, presence of arcs, etc.) and domain constraints (e.g., number of connections) to guide hypotheses selection and error correction. Kurtoglu and Stahovich [9] presented a similar approach that relies on physical reasoning to resolve ambiguities in sketches of mechanical devices and electronic circuits. In addition to enforcing consistency between pairs of connected components (e.g., a wire cannot connect to a bearing because one is an electrical device while the other is a mechanical one), it uses qualitative simulation to choose between multiple possible interpretations of the sketch. Their system avoids the parsing problem by requiring users to press a button to indicate that a symbol has been completed.

While these systems have demonstrated that domain knowledge can be used to overcome ambiguities, our approach differs in both the recognition process and the types of sketches considered. In order to interpret the intermixed drawing and handwriting found in molecular diagrams, our system employs both geometric features and template matching to parse and recognize symbols in the sketch. It also uses contextual hypothesis evaluation and automatic error correction to deal with the additional challenges in interpreting the spatial structure of chemical diagrams (e.g., aromatic rings, implicit chemical notations, sources of ambiguity shown in Figure 2-4).

6.2 Recognition Systems for Chemical Diagrams

There have also been efforts to recognize chemical sketches and diagrams. Tenneson and Becker [17] developed a sketch-based system that helps students visualize the three dimensional structure of an organic molecule. It avoids the ink parsing problem by requiring that all symbols be drawn using a single stroke, similar to the gesture based system described earlier. It also does not handle implicit structure such as omitted carbon and hydrogen atoms. Casey *et al.* [4] developed a system for extracting chemical graphics from scanned documents, but that work focused on printed chemical diagrams rather than freehand drawings. Also, unlike our system their approach did not handle non-planar chemical notations such as wedge and hash bonds.

Chapter 7

Discussion

7.1 Future Work

For our future work, we would like improve the performance of the system when dealing with messy sketches that contain over-traced symbols by expanding the image-based component of the symbol recognizer. We also plan to extend the system to recognize sketches of chemical reactions like the one in Figure 7-1.

While the system is currently aimed primarily at chemists and other researchers, we believe that it could also be used in chemistry education, for example as a learning aid in a high school chemistry classroom. As a student draws on a smart whiteboard, the system would provide real time feedback about improper bond structures, present a three dimensional visualization of the molecule, or animate a reaction. We are also currently working provide more real time feedback about the chemical compound as the user sketches, such as providing a continuously updated list of properties like atomic weight and automatically identifying parts of the molecule that may be toxic or dangerous.

7.2 Contributions

In this thesis we presented a sketch recognition system designed to interpret hand drawn chemical diagrams. We have shown that the system is capable of recognizing

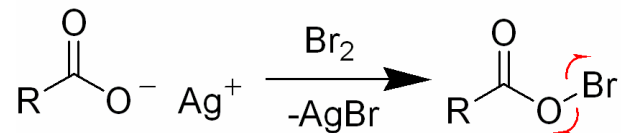


Figure 7-1: An example of an organic reaction.

common organic chemistry notations and is able to use domain knowledge to improve its interpretation of the sketch. Our system uses a trainable symbol recognizer to locate and identify symbols in the sketch. The recognizer incorporates both feature-based and image-based methods, allowing it to more reliably interpret messy sketches that contain a wide range of symbols, including intermixed text, drawing, and fill-in shapes. Analysis of the region surrounding each symbol allows the system to choose among competing interpretations and determine an initial structure for the molecule. Knowledge of chemistry, in particular atomic valence, then enables the system to check the validity of its interpretation and, if necessary, revise it to recover from errors.

Bibliography

- [1] C. Alvarado and R. Davis. SketchREAD: a multi-domain sketch recognition engine. In *UIST*, pages 23–32. ACM Press New York, NY, USA, 2004.
- [2] C. Bahlmann, B. Haasdonk, and H. Burkhardt. Online handwriting recognition with support vector machines – a kernel approach. In *IWFHR*, pages 49–54, 2002.
- [3] Chris Calhoun, Thomas F. Stahovich, Tolga Kurtoglu, and Levent Burak Kara. Recognizing multi-stroke symbols. In *AAAI Spring Symposium on Sketch Understanding*, pages 15–23, 2002.
- [4] R. Casey, S. Boyer, P. Healey, A. Miller, B. Oudot, and K. Zilles. Optical recognition of chemical graphics. *Document Analysis and Recognition*, pages 627–631, Oct 1993 1993.
- [5] Scott D Connell and Anil K Jain. Learning prototypes for on-line handwritten digits. *ICPR*, 01:182, 1998.
- [6] Microsoft Corporation. Microsoft windows xp tablet pc edition software development kit, 2005.
- [7] Leslie Gennari, Levent Burak Kara, and Thomas F. Stahovich. Combining geometry and domain knowledge to interpret hand-drawn diagrams. *Computers and Graphics*, 29(4):547–562, 2005.
- [8] Levent Burak Kara and Thomas F. Stahovich. Hierarchical parsing and recognition of hand-sketched diagrams. In *UIST*, pages 13–22, 2004.
- [9] T. Kurtoglu and T. Stahovich. Interpreting schematic sketches using physical reasoning. In *AAAI Spring Symposium on Sketch Understanding*, 2002.
- [10] L. Long and M. Rowe. Visual similarities of pen gestures. In *CHI*, 2000.
- [11] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998.
- [12] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

- [13] Dean Rubine. Specifying gestures by example. In *SIGGRAPH*, pages 329–337, New York, NY, USA, 1991. ACM Press.
- [14] Tevfik Metin Sezgin. Feature point detection and curve approximation for early processing of free-hand sketches. Master’s thesis, May 2001. Department of EECS, MIT.
- [15] Michael Shilman, Hanna Pasula, Stuart Russell, and Richard Newton. Statistical visual language models for ink parsing. *AAAI Spring Symposium on Sketch Understanding*, 2002.
- [16] Michael Shilman, Paul Viola, and Kumar Chellapilla. Spatial recognition and grouping of text and graphics. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, 2004.
- [17] Dana Tenneson and Sascha Becker. Chempad: Generating 3d molecules from 2d sketches. *ACM SIGGRAPH 2005 Posters*, 2005.
- [18] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.