Computers and the Humanities **34:** 377–392, 2000.
© 2000 *Kluwer Academic Publishers. Printed in the Netherlands.*

Using Constraint Logic Programming to Analyze the Chronology in "A Rose for Emily"

JENNIFER BURG[1], ANNE BOYLE[2] and SHEAU-DONG LANG[3]
[1]*Department of Computer Science, Wake Forest University, Winston-Salem, NC 27109 (E-mail: burg@mthcsc.wfu.edu);* [2]*Department of English, Wake Forest University, Winston-Salem, NC 27109 (E-mail: boyle@wfu.edu);* [3]*School of Computer Science, University of Central Florida, Orlando, Florida 32816, USA (E-mail: lang@cs.ucf.edu)*

Abstract. William Faulkner's non-chronological story telling style has long been a challenge to critics and a puzzle to beginning literature students. "A Rose for Emily," one of Faulkner's most frequently anthologized stories, exemplifies the complexity of Faulkner's treatment of time. In this paper, we apply a constraint-based problem solving method to an analysis of the chronology of "A Rose for Emily." Constraint logic programming is a declarative programming language paradigm that solves problems by enforcing constraints among variables. CLP's ability to sort numeric variables that do not yet have definite values makes it possible to sort the events of "A Rose for Emily" with only fragmented and relative time information. In attempting to sort the events of the story, we find an inconsistency in the temporal references scattered throughout the narrative. After removing this inconsistency, we are able to compare our chronology with earlier ones and discuss the thematic relevance of Faulkner's nonlinear plots.

Key words: chronology, constraint logic programming, constraints, Faulkner

But to what purpose
Disturbing the dust on a bowl of rose-leaves
I do not know. T.S. Eliot, "Burnt Norton"

1. Faulkner's Treatment of Time

One of the most intriguing and perplexing elements of William Faulkner's work is his treatment of time. Through shifts in narrators, non-chronological story telling, and reappearing characters, Faulkner's stories can leave unseasoned readers with a jumble of "incidents" related through the ramblings of memory. While a deeper understanding of Faulkner's nonlinear approach to time may come later, a first understanding of Faulkner's stories requires some sorting out of these events and their relative significance.

Faulkner's most widely read short story, "A Rose for Emily," illustrates this difficulty. Beginning with Emily's death, Faulkner meanders back and forth through a period of change from the pre-Civil War South to the modern age. From the very first words of the story's first paragraph – "When Emily Grierson died" – Faulkner weaves together the events of Emily's life using frequent references to time, telling us that some event occurred "thirty years before" this or "eight or ten years after" that, or on the "next night" or "within three days." All the pieces of the puzzle seem to be there, but it is not immediately clear how they fit together. In the nearly 70 years since the story's publication, critics have offered numerous chronologies, defended on the basis of biographical, historical, contextual, textual, or canonical evidence (Going, 1958; Woodward, 1966; Hagopian, 1964; McGlynn, 1969; Nebeker, 1970, 1971; Wilson, 1972; Perry, 1979; Schwab, 1991). (See "A Comparison of Chronologies" below.) In this paper, we describe a computer-based tool for comparing and checking the consistency of chronologies, and show how application of this tool can be pedagogically and analytically useful. Applying a constraint-based sorting procedure to the events of "A Rose for Emily," we uncover an inconsistency in the relative time information given in the story. Removing the inconsistency, we then sort the events and compare our results with earlier interpretations. We conclude that while a preoccupation with chronological ordering would be a misreading of Faulkner's "huge meadow" of time, testing the consistency of various proposed chronologies is useful to an understanding of his characters' motivations and historical circumstances.

2. A Chronology of "A Rose For Emily"

Let us begin by summarizing "A Rose for Emily," not in Faulknerian style, but straight ahead from beginning to end.

Emily Grierson was the only daughter of a once-prominent family of the Old South. She appeared to have many suitors in her youth, but for some reason Emily never married. Maybe she was too much under the control of her domineering father. Maybe the Griersons held themselves a little too high above the rest of the town, and none of the suitors were considered good enough for Emily. Maybe none of the suitors ever actually asked Emily to marry him. In any case, when Emily's father died, she apparently could not accept the loss and denied his death for three days, until the townspeople finally convinced her to bury him. Left with almost nothing and still not married by the age of thirty, Emily found herself an object of pity rather than envy among the townspeople. Then Homer Barron, a construction worker, came to town. Before long, Emily was seen everywhere with Homer. He wasn't the kind of man the townspeople would have expected Emily to marry – a common laborer and a Yankee – and the propriety of her unchaperoned relationship with him was questionable. Emily's out-of-town cousins were called in to save Emily's reputation. While they were in town, Emily went to the druggist and bought some arsenic, and everyone feared that she would try to kill herself. But then she bought a man's toilet set with the initials H.B. on each piece, and it looked like a wedding was in the offing. The wedding never materialized.

Homer disappeared for three days, returned, and then was never seen again. Emily became reclusive after that. She was sometimes seen through the window, like the time when some men were sent out to sprinkle lime around her house because it smelled so bad. For about eight years, Emily gave China painting lessons to earn money. Once, a deputation of town aldermen was sent to her house to tell her that she had to pay taxes. She refused and sent the men to see Colonel Sartoris – a man who had been dead for quite some time. But other than that, no one but her manservant entered her house. On the day of her funeral, the townspeople finally got to see the inside of her house again. This is when they found out what happened to Homer Barron. When they entered Emily's bedroom, there lay Homer's skeleton on the bed. Next to it, on the indentation of a pillow, was a long strand of grey hair.

But we've spoiled the fun here. Faulkner does not tell this story in so straightforward a fashion. Instead, he sends the reader on a dizzying voyage by referring to specific moments in time that have no central referent, and thus he weaves the past into the present, the present into the past. This nonlinearity has thematic significance to the story, as we shall see later. But beginning readers of the story are generally most concerned with a basic understanding of what happened and when, and they naturally pay close attention to the many clues regarding time. At the time of Emily's death, no one "had seen [the inside of her house] for at least ten years." "In 1894 ... Colonel Sartoris ... remitted her taxes ...." A deputation of aldermen from "the next generation" then called on her to tell her the deal was off. At that time "Colonel Sartoris had been dead almost ten years." "She had vanquished their fathers thirty years before about the smell." And so forth. The time information is abundant, but how are we to sort these events with some confirmation that our sorting is consistent?

3. Sorting Points in Time with Constraint Logic Programming

The answer to the chronology problem suggested itself to us in the form of a programming language paradigm in which problems are expressed in terms of constraints. *Constraint logic programming* (CLP) is a family of programming languages based upon declarative rather than procedural problem descriptions. In CLP, a program is a statement of what must be true of all the variables in the problem for the solution to be attained. For example, a simple CLP program could describe a family relationship.

```
parent(X,Y):-
        father(X,Y).
parent(X,Y):-
        mother(X,Y).
mother(ellen, billy).
mother(margie, amanda).
```

```
father(jim,jonathan).
father(stan,timothy).
```

This program states that *X* is the parent of *Y* if *X* is the mother of *Y*, or *X* is the parent of *Y* if *X* is the father of *Y*. It then asserts a few facts about who is the father or mother of whom.

In CLP languages, problem statements can be augmented with the expression of constraints in domains such as integers, rational numbers, or Boolean values. Prolog is the best known language in this family. CLP(*R*), the language we apply to the "Emily" chronology, is even more powerful than Prolog, offering constraint satisfaction in the real-number domain. In CLP(*R*), a program can include equations and inequalities that express constraints among the program variables, which can take on real-number values (Cohen, 1990; Colmerauer, 1990). For example, in CLP we might state that if two sets of parents have children close to the same age, they picnic together.

```
picnic_together(A,B):-
        parent(A,X),
        parent(B,Y),
        age(X,AgeX),
        age(Y,AgeY),
        close_in_age(AgeX,AgeY).
close_in_age(AgeX,AgeY):-
        AgeX-AgeY <= 3.
close_in_age(AgeX,AgeY):-
        AgeY-AgeX <= 3.
age(amanda,12).
age(billy,10).
Etc ...
```

Applied to "A Rose for Emily, constraints can be used to express the time information given by Faulkner, where variables represent points in time to be sorted. Part of the power of CLP(*R*) lies in its ability to verify that values are possible for all program variables in the context of existing constraints without requiring that the variables have *specific* values assigned to them. This makes it possible to apply CLP(*R*) to the sorting problem. In an ordinary procedural programming language such as C or Pascal, sorting is done on specific values that have inherent ordering and can be judged as greater than, equal to, or less than each other in some sense – numbers or letters of the alphabet, for example. But what if we wish to sort variables whose values are unknown at the time of the sort – points of time about which we have only partial information regarding their relative positions? With CLP(*R*), we can express time-related information in the form of equations and inequalities where points in time are either constants or variables.

Table I. Variables and their meaning

Variable	Meaning
A	Emily's death
B	Last time anyone but Emily's manservant saw the inside of her house
C	Colonel Sartoris remitted Emily's taxes
D	A deputation called on Emily asking her to start paying taxes again
E	Emily stopped giving China painting lessons
F	Colonel Sartoris died
G	There was a bad odor around Emily's house
H	Emily's father died
I	Homer Barron came to town
J	Homer disappeared
K	Emily was born
L	Emily appeared in town again after Homer's disappearance
M to N	First period of time when Emily shut her doors to the public
N to E	Period of China painting lessons
E to A	Second period of seclusion

For example, the fact that Emily's taxes were remitted in 1894 is expressed as $C = 1894$; and the fact that, with the exception of Emily's manservant, no one had entered her house for at least 10 years prior to her death becomes $A - B \geq 10$. These points in time can then be sorted, perhaps with more than one ordering possible.

Thus our first step is to express the time information in "Emily" in the form of equations and inequalities. We can then attempt to sort the variables, each of which represents a specific event. Table I below gives the variables to be used in the time constraints along with their meanings. Table II gives the constraints, with the corresponding information from which each constraint was inferred. Figure 1 gives the CLP(R) program, consisting of the constraints followed by a *sort* predicate. The sorting procedure places the time points in all possible consistent orderings given the constraints on their relative positions.

4. Constraint Satisfaction and Inconsistency

In CLP programming, problems are presented declaratively, and the underlying system finds a solution to the problem (i.e., program) through a combination of resolution-based inference and constraint satisfaction. As execution proceeds, a constraint "solver" continually checks the satisfiability of the system of constraints it has encountered in the program so far. If the constraints are satisfiable, it is possible to assign values to all the variables such that all the constraints remain

Table II. Constraints and their meaning

Constraint	Meaning
$A - B \geq 10$	No one besides the manservant had seen the inside of Emily's house for at least 10 years before her death.
$C = 1894$	Colonel Sartoris remitted Emily's taxes in 1894.
$D - C \geq 10$ $D - C \leq 20$	A generation later, a deputation called on Emily to tell her that she would have to pay taxes after all.
$D - E \geq 8$ $D - E \leq 10$	Eight or ten years passed between the time when Emily last gave China painting lessons and the time the deputation called on her.
$F > C$	Colonel Sartoris died after he remitted Emily's taxes.
$D - F > 9$ $D - F < 10$	Colonel Sartoris died almost 10 years before the deputation called on Emily.
$D - G = 30$	There was a bad odor around Emily's house 30 years before the deputation called on her about her taxes.
$B \geq D$	The last time anyone but the manservant saw the inside of Emily's house had to be after or at the same time as the visit of the deputation.
$B \geq E$	The last time anyone but the manservant saw the inside of Emily's house had to be after or at the same time as the last China painting lesson.
$G - H = 2$	The odor around Emily's house appeared two years after her father's death.
$I > H$	Homer Barron came to town after Emily's father died.
$I - K > 30$	Emily was older than 30 when Homer Barron came to town.
$J > I$	Homer disappeared after he came to town.
$G - J < 0.5$ $G > J$	The odor appeared less than 6 months after Homer disappeared.
$L - J < 0.5$	Emily appeared again on the streets after Homer's disappearance.
$A - K = 74$	Emily died at the age of 74.
$M > L$	The first period when Emily shut her doors to the public happened after her reappearance after Homer's death.
$E - N \geq 6$ $E - N \leq 7$	Emily gave China painting lessons for 6 or 7 years.
$E = B$	When Emily shut the door on her last China painting student, no one but her manservant saw the inside of her house after that.
$N - K \leq 45$ $N - K \geq 38$	Emily was about 40 when she gave China painting lessons.
$C \geq N$ $C \leq E$	Colonnel Sartoris remitted Emily's taxes while the China painting lessons were going on.
$M < N$	The beginning of the period of seclusion has to be before the end.
$A > E$, $A > K$, Etc.	Emily died after everything else.

```
emily([A,B,C,D,E,F,G,H,I,J,K,L,M,N], LIST):-
      A-B ≥ 10,
      C = 1894,
      D-C ≥ 10, D-C ≤ 20,
      D-E ≥ 8, D-E ≤ 10,
      F > C,
      D-F > 9, D-F < 10,
      D-G = 30,
      B ≥ D, B ≥ E,
      G-H = 2,
      I > H,
      I-K > 30,
      J > I,
      G-J < 0.5,
      G > J,
      L-J = 0.5,
      A-K = 74,
      M > L,
      E-N ≥ 6, E-N ≤ 7,
      %E = B,        /*This constraint inserts a conflict*/
      N-K ≤ 45,
      N-K ≥ 38,
      C ≥ N, C ≤ E,
      M < N,
      A > E, A > K,
      insort([A,B,C,D,E,F,G,H,I,J,K,L,M,N], LIST).

insort([],[]).
insort([X|L],M):-
      insort(L,N), insortx(X,N,M).
insortx(X, [A|L], [A|M]):-
      A ≤ X, insortx(X, L, M).
insortx(X,L,[X|L]):-
      X ≤ A, starts(A,L).
insortx(X,[],[X]).
starts(A, [A|L]).
```

Figure 1. The CLP(R) program.

true statements. If the constraints are unsatisfiable, the program fails, since a solution is impossible. This leads us to a second advantage of applying CLP(*R*) to the chronology problem – that is, its ability to uncover conflicts in constraint systems, and to pinpoint the cause of the conflict. With some additional bookkeeping done in program execution, it is possible to determine a minimal set of inconsistent constraints in the case where a conflict in the constraint set is detected.[1] (By definition, if just one of the constraints is deleted from a minimal conflict set, the set is no longer inconsistent.) Stated in terms of the "Emily" chronology, if the time information given by Faulkner is inherently inconsistent (as presented to the CLP(*R*) program), the program will detect the inconsistency and report back a subset of the time constraints that could not possibly be true simultaneously.

Finding an inconsistency was not our original purpose in applying CLP to "A Rose for Emily." But an inconsistency did in fact appear. If you attempt to run the program given in Figure 1, you'll find that it never actually gets to the *sort* procedure. This is because the constraint solver discovers a conflict when it reaches the constraint $E = B$. The CLP(*R*) interpreter lists the conflicting constraints as follows:

$$B \geq D$$
$$D - E \geq 8$$
$$E = B$$

Intuitively, it should be clear that these constraints are irreconcilable. Faulkner tells us that the deputation of aldermen went *into* Emily's house:

> A deputation waited upon her, knocked at the door through which no visitor had passed since she ceased giving china-painting lessons eight or ten years earlier. They were admitted by the old Negro into a dim hall from which a stairway mounted into still more shadow.

This we represent as $D - E \geq 8$ (and $D - E \leq 10$). Common sense then dictates that the last time anyone saw the inside of Emily's house (moment B) was *at the same time or after* the deputation's visit, i.e., $B \geq D$. In the same passage we learn that the china-painting lessons occurred between eight and ten years earlier. Later in the story we are told that the china-painting students were the last townspeople to enter Emily's house:

> Then the newer generation became the backbone and the spirit of the town, and the painting pupils grew up and fell away and did not send their children to her with boxes of color and tedious brushes and pictures cut from the ladies' magazines. The front door closed upon the last one and remained closed for good.

This we translate into the constraint $E = B$.

One way to reconcile this conflict is to assume that when Faulkner says that "the front door closed …for good," he means that only in the context of the

Table III. The timeline for “A Rose for Emily”

#1	K	H	I	J	G	L	M	N	1894 (C)	F	E	D	B	A
#2	K	H	I	J	G	L	M	N	1894 (C)	F	E	B	D*	A
#3	K	H	I	J	G	L	M	1894 (N)	1894 (C)	F	E	D	B	A
#4	K	H	I	J	G	L	M	1894 (N)	1894 (C)	F	E	B	D*	A
#5	K	1872 (H)	I	J	1874 (G)	L	M	N	1894 (C)	1894 (E)	F	1904 (D)	B	A
#6	K	1872 (H)	I	J	1874 (G)	L	M	N	1894 (C)	1894 (E)	F	1904 (D)	1904 (B)	A
#7	K	H	I	J	G	L	M	N	1894 (C)	E	F	D	B	A
#8	K	H	I	J	G	L	M	N	1894 (C)	E	F	B	D*	A
#9	K	H	I	J	G	L	M	1894 (N)	1894 (C)	E	F	D	B	A
#10	K	H	I	J	G	L	M	1894 (N)	1894 (N)	E	F	B	D*	A

*Note that B = D in this sorting.

China painting lessons. Or maybe the deputation of aldermen simply went in the back door. In any case, *some* re-interpretation is necessary in order to arrive at a consistent chronology. Here we see that CLP can either help to clarify our own possible misreadings of the story, or point up an inconsistency in the chronology that may have escaped our notice, despite many readings, because of the nonlinear presentation of events.

Let’s assume for now that our interpretation of the individual pieces of time information was correct, but the chronology is inconsistent. Since the conflict set uncovered by our CLP(R) interpreter is a *minimal* conflict set, we can resolve it by removing any one of the constraints. In this case, we can remove the constraint $E = B$, and it is then possible to sort the events of the story.

The complete CLP(R) program is given in Figure 1. With the constraint $E = B$ left in, the solver never executes the sort because it finds a conflict in the constraint set. With the constraint $E = B$ deleted, the solver finds one feasible solution (disregarding duplicate solutions that swap the position of time points that can be equal). Not all variables are given specific values by the sort, but their relative positions are determined.

The ten possible sortings are given in Table III. They vary in the following ways.

- The remission of taxes is either at the beginning of Emily's period of china-painting lessons, in the middle of this period, or at the end of this period. (Either C is the same year as N, it is between N and E, or it is in the same year as E.)
- Colonel Sartoris dies either during Emily's period of giving china-painting lessons, or after this period (either $F \leq E$ or $F > E$).
- The members of the deputation requesting that Emily pay taxes are the last townspeople (other than Emily's servant) to see the inside of her house until her death ($B = D$), or someone else is the last person to see the inside of Emily's house ($B \geq D$).

This sorting exercise and an examination of the resulting timeline can be helpful to students trying to understand Faulkner's work, for through the sorted timeline we see more clearly the transitional time period during which the events of the story take place. In our sample timeline, Emily was born in 1850 and died in 1924, her life beginning before the Civil War and ending within America's period of industrialization and growth. Considering the different possible historical settings in which Emily lived makes an excellent starting point for classroom discussions.

5. A Comparison of Chronologies

CLP gives us an objective method for comparing the various chronologies proposed for "Emily" over the years. Moore (1992) divides the chronologies into two groups based on the position of the one explicit date in the timeline – 1894, the year when Emily's taxes were remitted. The first group, including Woodward, McGlynn, Nebeker, and Wilson, set the remission of taxes in the same year as the death of Emily's father. The second group – Going, Hagopian et al., Nebeker (revised), Brooks, and Perry – set the remission of taxes at the time of the china-painting lessons.

Clearly, our chronology falls into the second group. To determine the earliest birthdate possible for Emily within our constraints, we can insert an equation experimentally inserting a specific birthdate into the constraint set. By this means, we find that the time constraints don't accommodate a birthdate earlier than 1842 ($K = 1842$) or later than 1856.

After testing other specific dates similarly, we propose the timeline given in Table IV. Interestingly, our sorting program confirms a possible birthdate of 1850, lending credence to one of the earliest chronologies – that of Going (1958). This chronology is further reinforced by the date assigned to the writing of "A Rose for Emily" in Malcolm Cowley's Viking Portable edition of Faulkner's works (Cowley, 1946). It seems reasonable that Faulkner would set the year of Emily's death at the writing of the story – 1924 – and probably not after. Faulkner gives

Table IV. The timeline for “A Rose for Emily”

K (1850)	Emily is born
H (1879)	Emily’s father dies
I	Homer Barron comes to town
J	Homer disappears
G (1881)	A bad odor appears around Emily’s house
L	Emily reappears after a period of seclusion
M	Emily begins a second period of seclusion
N (1894)	Emily ends second period of seclusion; begins giving China painting lessons
C (1894)	Emily’s taxes are remitted
E (1901)	Colonel Sartoris dies
F	Emily stops giving China painting lessons
D (1911)	A deputation of town officials call on Emily about her taxes
B (1914)	Last time anyone but Emily’s servant sees the inside of her house
A (1924)	Emily dies at the age of 74

Emily’s age at her death to be 74, which would put the date at 1924 in our chronology.

Moore also points out that in an early manuscript of “Emily,” the 1894 remission of Emily’s taxes was described instead as “that day in 1904 when Colonel Sartoris . . . remitted her taxes dating from *the death of her father 16 years back*, on into perpetuity” (emphasis ours). To check the plausibility of this 1904 date within our interpretation, we substitute $C = 1904$ for $C = 1894$ and again try to sort the time points. The sort fails, indicating that Faulkner perhaps was correcting his own inconsistency. However, if we leave the tax remission at 1894, we find that Emily’s father indeed could have died 16 years before the date of the tax remission, as Faulkner originally stated. (We can check this by inserting $H = 1878.1$ into our program. $H = 1878$ won’t work, but 15.9 years could certainly be considered 16 years to Faulkner’s degree of accuracy!)

6. Relaxing our Constraints in the Meadow of Time

And so, you might ask, “to what purpose” have we “disturbed the dust on a bowl of rose-leaves”? Faulkner is well-known for twisting chronology “almost beyond recognition” (Sullivan, 1971), not only in “A Rose for Emily,” but in much longer, more complex novels. There are many hints to the readers that we should not be too strict-minded about linear time. Consider, for example, Faulkner’s obscure and complicated masterpiece on time and narration, *The Sound and the Fury*, published just one year before “A Rose for Emily.” On the day he will commit suicide, Faulkner’s Quentin Compson, perhaps trying to reverse time and his sister's fall from virginity, breaks his grandfather's watch, recalling his father's admonition that "time is dead as long as it is being clicked off by little wheels; only when the clock stops does time come to life." Perhaps readers and critics of "A Rose for Emily" should heed Mr. Compson's advice and strive to "forget [time] now and then for a moment and not spend all your breath trying to conquer it."

Our investigation into the apparent slip in Faulkner's chronology reinforces the view that Faulkner's fictional world in "A Rose for Emily" is not to be laid out in linear time, but exists out-of-step with time as most of us know it. "[T]ime comes to life when the clock stops" – and Emily, who, like Quentin, cannot accept loss or a diminished world, struggles mightily throughout the story to stop the clock. Indeed, Emily, has always denied time and change. She can't accept the death of her father or of Colonel Sartoris; she rejects the end of the old order of Southern life and ignores "the next generation, with its more modern ideas"; finally, she refuses to lose her beau. Thus, she cuts herself off from time and constructs a room in which she stops the clock for Homer Barron. Not only does she murder him, but she continues to sleep for years beside the body of a lover who cannot betray her.

As the narrator who tells the story reminds us, there are two ways of understanding time. Some see it as a "mathematical progression," a fixed and coherent chronology. Emily prefers to view time as "not a diminishing road but, instead, a huge meadow which no winter ever quite touches ...." While Faulkner teases readers to imagine that they can follow time through that diminishing road into the past and arrive at a coherent vision, what they find is inconsistency in time and of motive. It may be that to understand Emily, we must give up our orderly sorting of experience, and for a moment view time as "an undying meadow," a place untouched by death and loss.

This is not to say that a study of Faulkner chronologies is a pointless one. Faulkner's stories and novels are deeply rooted in their historical setting. It certainly makes a difference to understand that Emily's spinsterhood coincided with the diminished glory of the Old South, that she denied the death of her father for days and her lover for years, and that she lived until 1924, a time vastly different from the pre-Civil War days of her youth. CLP provides an objective means by which we can compare chronologies, check for our own misreadings, find possible inconsistencies, and consider their thematic repercussions. It also offers a tool with which beginning literature students can make sense of a story told in Faulkner's unconventional style, giving them a way to sort out the plot in a variety of timelines, leading them inevitably to deeper discussions of the text. For more complex works, this experimentation with sorting could be even more revealing. It would be interesting to see how events in *The Sound and the Fury* might ....

But let us leave that to another time.

Appendix

The Constraint Satisfaction Algorithm and Conflict Sets

In what follows, we assume the reader has a basic knowledge of Prolog-like languages, and refer the reader to [Clocksin and Mellish] and [Sterling and Shapiro] for details.

The execution of a CLP(*R*) program can be divided into two components: resolution-based inferencing coupled with unification for the binding of variables, in the manner first applied to Prolog [Kowalski; Robinson]; and constraint satisfaction in the real-number domain [Colmerauer; Cohen]. Thus, a CLP(*R*) interpreter can be divided into two corresponding components: an inference engine to perform a depth-first search through the program space; and a constraint solver to check the satisfiability of the collected constraints at the entrance to each predicate.

The basic implementation of a constraint solver for CLP(*R*) is given in [Jaffar et al.], to which we have added mechanisms for conflict identification and intelligent backtracking [Burg, Lang, and Hughes]. The task of the solver is to determine if the constraints collected thus far during execution are satisfiable. That is, given the bindings performed during unification, is it possible to assign real-number values to the remaining program variables such that all constraints simultaneously hold true? (We should note that in both the *Jaffar et al.* implementation and our own, the constraints are limited to linear equations and inequalities for efficiency reasons). The constraints arise from one of two sources – either from arithmetic expressions (containing variables) that are equated during unification, or from equations or inequalities in the bodies of program clauses. Since more constraints are added to the system each time a clause is entered during program execution, constraint satisfaction is necessarily incremental.

More formally, the solver's problem is as follows: Say that for each inequality constraint

$$a_1x_1 + \ldots + a_nx_n \leq b \text{ (or, } a_1x_1 + \ldots + a_nx_n \geq b)$$

the inequality is converted to an equation of the form

$$a_1x_1 + \ldots + a_nx_n + 2 = b \text{ (or, } a_1x_1 + \ldots + a_nx_n - s = b)$$

where the slack variable s is assumed to have a non-negative value. (This can be easily generalized to strict inequalities as well, but we will not do so here for simplicity of notation.) At any moment in execution, we can assume we have a satisfiable system of the form

$$M \cdot X = B \qquad (1)$$

where $M = \begin{bmatrix} 1 & * & \ldots & \ldots & * \\ 0 & 1 & \ldots & \ldots & \ldots \\ \ldots & \ldots & 1 & \ldots & \ldots \\ 0 & \ldots & 0 & 1 & * \end{bmatrix}$ is an $m \times n$ matrix $(m \leq n)$

for m equations and n unknowns, $x = \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix}$

represents the n unknowns, and $B = \begin{bmatrix} b_1 \\ \dots \\ b_m \end{bmatrix}$ represent the constants on the right-hand side of the equations, with all $b_i \geq 0$.

Notice that the unknowns $x_1, \dots, x_m$, corresponding to the 1's on the diagonal of the matrix M in (1), are the basic variables. The system (1) is satisfiable because an obvious solution is obtained by letting the basic variables $x_i = b_i$ for $1 \leq i \leq m$, and letting the non-basic variables $x_{m+1} = \dots = x_n = 0$.

Given such a system, then time a new constraint is encountered during program execution, the solver's problem is to check the satisfiability of the new system, including the new constraint.

In [Burg, Lang, and Hughes], we describe an incremental version of Gaussian elimination combined with the simplex method for checking the satisfiability of the system of constraints. All operations in this algorithm are basic row operations where a multiple of one row (i.e., constraint) is added to another as we determine which variable to "solve for" in each constraint. In effect, we keep substituting variables out of the last row of our constraint system until we arrive at one of two results: If the system (including the new constraint) is indeed satisfiable, then either the new constraint is completely eliminated because it is redundant, or the system is transformed into the same form as in (1), with one extra row added. If, on the other hand, the system is unsatisfiable when the new constraint is added to it, then our row operations has put the last row in the form

$$a_{m+1,m+1}x_{m+1} + \dots + a_{m+1,n}x_n = b_{m+1}$$

where $a_{m+1,i} \leq 0$ and $b_{m+1} > 0$. This last equation clearly cannot be satisfied by the unknown $x_i \geq 0$ for $1 \leq i \leq n$, because each coefficient $a_{m+1,i} \leq 0$ but the constant $b_{m+1} > 0$.

To identify a set of conflicting constraints in the system, we need to keep a record of the row operations performed during Gaussian elimination and the simplex procedure. Specifically, if we let matrix R represent the set of equations in their original form, each time a row operation is applied to an equation in R, the matrix R is transformed into $E \cdot R$, where E is a matrix corresponding to the row operation. Therefore, we could use a matrix B which is equal to the product of these E matrices to record the successive row operations. As a result, the current coefficient matrix M in the transformed system (as in (1)) is related to the matrix R by the simple equation

$$M = B \cdot R$$

It has been proved in [Burg, Lang, and Hughes] that the indices of the non-zero entries in the last row of matrix B identify exactly the rows (i.e., constraints) in R which cause the conflict. We have shown that the conflict set revealed by our intelligent backtracking solver is in fact a minimal conflict set – that is, if we remove any one constraint from the set, it is no longer inconsistent. (However, we should note that the conflict set revealed by the algorithm is not necessarily unique.)

Note

[1] More precisely, we have shown in earlier work that when standard Gaussian elimination and the simplex method are used to check the satisfiability of the constraints, a minimal conflict set can be identified directly. This information can be used as the basis of intelligent backtracking in the solution of CLP(R) problems [Burg, Lang, and Hughes]. The first implementation of CLP(R) emerged from a research team at Monash University in Australia, and from there it evolved to a compiled version released by IBM's Thomas Watson Research Center [Jaffar et al.]. Neither of these versions of CLP(R) had the conflict-identification feature described above. We have implemented our own constraint solver with conflict-identification and intelligent backtracking, and it is this CLP(R) implementation that we have applied to the "Rose for Emily" chronology problem. See the Appendix for details of constraint satisfaction and minimal conflict sets in CLP(R).

References

Burg, J., S.-D. Lang and C.E. Hughes. "Intelligent Backtracking in CLP(R)". *Annals of Mathematics and Artificial Intelligence* 17 (1996), 189–211.

Clocksin, W.F. and C.S. Mellish. *Programming in Prolog*, 3rd ed. New York: Springer-Verlag, 1987.

Cohen, J. "Constraint Logic Programming Languages". *Communications of the ACM* 33(7) (1990), 52–68.

Colmerauer, A. "An Introduction to Prolog III". *Communications of the ACM* 33(7) (1990), 69–90.

Cowley, M. "Introduction". *The Portable Faulkner*. New York: Viking, 1946, pp. 1–24.

Faulkner, W. *Collected Stories of William Faulkner*. New York: Random, 1950.

Faulkner, W. *The Sound and the Fury*. (1929) The Corrected Text. New York: Vintage, 1990.

Going, William T. Chronology in Teaching "A Rose for Emily." Reprinted in Inge, 76–83.

Hagopian, John V., W.G. Cunliffe and M. Dolch. " 'A Rose for Emily' " Reprinted in Inge, 76–83.

Inge, M. Thomas. *William Faulkner: A Rose for Emily*. The Charles Merrill Literary Casebook Series. Columbus, Ohio: Merrill, 1970.

Jaffar, et al. The CLP(R) Language and System. *ACM Transactions on Programming Languages and Systems* 14(3) (July 1992), 339–395.

Kowalski, R. "Algorithm = Logic + Control". *Communications of the ACM* 22(7) (1979), 424–436.

Littler, Frank. "The Tangled Thread of Time: Faulkner's 'A Rose for Emily.' " *Notes on Mississippi Writers* 14(2) (1982), 80–86.

McGlynn, P.D. The Chronology of "A Rose for Emily." Reprinted in Inge, 90–92.

Moore, G.M. "Of Time and its Mathematical Progression: Problems of Chronology in Faulkner's 'A Rose for Emily.' " *Studies in Short Fiction* 29 (1992), 195–204.

Nebeker, H.E. "Emily's Rose of Love: Thematic Implications of Point of View in Faulkner's 'A Rose for Emily.' " *Bulletin of the Rocky Mountain Modern Language Association* 24 (1970), 3–13.

Nebeker, H.E. "Chronology Revised". *Studies in Short Fiction* 8 (1971), 471–473.

Perry, M. “Literary Dynamics: How the Order of a Text Creates its Meanings” [With Analysis of Faulkner’s “A Rose for Emily”]. *Poetics Today* 1(1–2) (Autumn 1979), 35–64, 311–361.

Sullivan, R. “The Narrator in ‘A Rose for Emily.’ ” *Journal of Narrative Technique* 1 (1971), 159–178.

Robinson, J.A. *Logic and Logic Programming. Communications of the ACM* 35(3) (1992), 40–64.

Schwab, Milinda. “A Watch for Emily”. *Studies in Short Fiction* 28(2) (1991), 215–217.

Sterling, L. and E. Shapiro. *The Art of Prolog: Advanced Programming Techniques*, 2nd ed. Cambridge: MIT Press, 1994.

Van Hentenryck, P. *Constraint Satisfaction in Logic Programming*. MIT Press, 1989.

Wilson, G.R., Jr. “The Chronology of Faulkner’s ‘A Rose for Emily’ Again”. *Notes on Mississippi Writers* 5 (Fall 1972), 56, 58–62.

Woodward, R.H. The Chronology of “A Rose for Emily”. Reprinted in Inge, 84–86.