

Closures and Algorithms

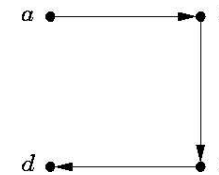
1 2 3 4 5 6 7 8 9 10 11 12 13 14



Closures and Algorithms

Closures

Consider the relation $R = \{(a,b), (b,c), (c,d)\}$ on the set $\{a, b, c, d\}$. This relation is not reflexive, symmetric or transitive.

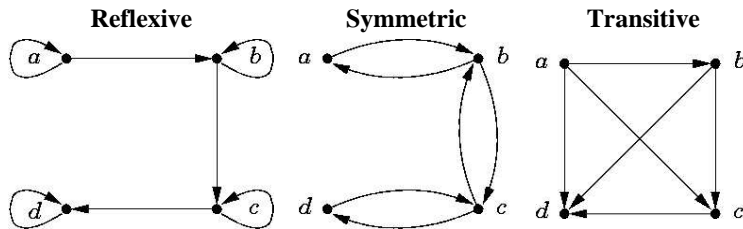


1 2 3 4 5 6 7 8 9 10 11 12 13 14

Closures and Algorithms

Closures

We can, however, create new relations from R that have each of these properties:



1 2 3 4 5 6 7 8 9 10 11 12 13 14

Closures and Algorithms

Closures

These new relations are called **closures** because they introduce the necessary elements into a relation to make it have certain properties; ones that cause the relation to be *closed* in some sense.

Consider the integers. This set is **closed under the operation of addition** (or just **closed under addition**). By this we mean that the relation

$$S = \{(a, b, c) \mid a, b, c \text{ are integers and } c = a+b\}$$

is closed. All this says is that the sum of two integers is another integer. The relation S here is a set of 3-tuples, the last entry of which is the sum of the first two. *Every possible 3-tuple where the last integer is the sum of the first two integers is in S .* Thus S is closed, and we say that \mathbf{Z} is closed under addition.

1 2 3 4 5 6 7 8 9 10 11 12 13 14

Closures and Algorithms

[prev](#) | [slides](#) | [next](#)

Closures

Returning to the relation R we began with, let R^* be the **reflexive transitive closure of R** . Thus R^* is obtained by introducing the ordered pairs that are necessary to create a reflexive and transitive relation from R .

Thus

$$R = \{(a,b), (b,c), (c,d)\}$$

$$R^* = \{(a,a), (a,b), (a,c), (a,d), (b,b), (b,c), (b,d), (c,c), (c,d), (d,d)\}$$

1 2 3 4 5 6 7 8 9 10 11 12 13 14

Closures and Algorithms

[prev](#) | [slides](#) | [next](#)

Algorithms

Saying that " R^* is obtained by introducing the ordered pairs that are necessary to create a reflexive and transitive relation from R " is fine and well, but how exactly is this done? We want to describe an **algorithm** that does this for us.

An **algorithm** is a sequence of steps that produce a desired result. Algorithms have the following properties:

- they have input,
- they produce output,
- they are definite and unambiguous,
- they will always terminate.

In addition, it is usually desirable that an algorithm not only terminates, but that it does so in a reasonable amount of time.

1 2 3 4 5 6 7 8 9 10 11 12 13 14

Closures and Algorithms

[prev](#) | [slides](#) | [next](#)

Closures

We will see several forms of closures during this course, some of which will apply to alphabets and languages.

The key thing to remember about them is that the closure of a relation creates a (possibly) new relation in which all n -tuples are present that are required for the (new) relation to have some particular property.

1 2 3 4 5 6 7 8 9 10 11 12 13 14

Closures and Algorithms

[prev](#) | [slides](#) | [next](#)

Algorithms

Let $A = \{a, b, c, d\}$ and let $R = \{(a,b), (b,c), (c,d)\}$ be a relation on A . Notice that $R \subseteq A \times A$, as will be R^* , the reflexive transitive closure of R .

The following algorithm takes R and A and produces R^* . In the algorithm $n = |A|$.

$$R^* := \emptyset$$

for $i = 1, \dots, n$ do
 for each $(b_1, b_2, \dots, b_i) \in A^i$ do
 if (b_1, b_2, \dots, b_i) is a path in R then add (b_1, b_i) to R^*

Note: A^i is the Cartesian product of A with itself $i-1$ times: $A^3 = A \times A \times A$ and is a set containing i -tuples.

1 2 3 4 5 6 7 8 9 10 11 12 13 14

Closures and Algorithms

[prev](#) | [slides](#) | [next](#)

Algorithms

This algorithm has clearly defined input and the steps are unambiguous and clear. That this algorithm is correct is fairly easy to see.

Initially R^* is empty. Ordered pairs are added to it only when a path from the first element to the second element has been found. This includes reflexive paths and ensures that all transitive paths are represented. Ordered pairs are not added to R^* for any other reason. Finally, every possible path is considered, so we are sure that none are overlooked. Thus, the algorithm is correct and therefore produces the desired output.

The algorithm terminates since $n = |A|$ is a finite number and the outer loop is done n times. The inner loop is also finite since $|A^n|$ is finite (but possibly quite large). Therefore we know that the algorithm will eventually terminate.

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#)

Closures and Algorithms

[prev](#) | [slides](#) | [next](#)

Algorithms

Unfortunately, algorithms that are $O(n^n)$ are not very useful for large values of n . Assume that we have a computer that can perform 100 million path checks per second. The table on the next slide shows how long it would take to form the reflexive transitive closure of relations on sets of up to 20 elements.

Note: Actually the table assumes that the number of steps for this operation is n^n , not

$$n + n^2 + n^3 + \dots + n^n$$

Thus the table actually underestimates the time required.

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#)

Closures and Algorithms

[prev](#) | [slides](#) | [next](#)

Algorithms

Just how many steps are required for this algorithm?

Let's consider a step to be answering the question as to whether a given i -tuple represents a path in R . In this case the first time through the outer loop there are n steps. The second time through the outer loop there are n^2 steps. Continuing in this fashion we see that when all n outer loops have been completed that we will have done

$$n + n^2 + n^3 + \dots + n^n$$

steps. This is $O(n^n)$.

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#)

Closures and Algorithms

[prev](#) | [slides](#) | [next](#)

Algorithms

n	Seconds	Years	n	Seconds	Years
1	1.000000e-08	3.168809e-16	11	2.853117e+03	9.040981e-05
2	4.000000e-08	1.267524e-15	12	8.916100e+04	2.825342e-03
3	2.700000e-07	8.555784e-15	13	3.028751e+06	9.597533e-02
4	2.560000e-06	8.112150e-14	14	1.111201e+08	3.521182e+00
5	3.125000e-05	9.902527e-13	15	4.378939e+09	1.387602e+02
6	4.665600e-04	1.478439e-11	16	1.844674e+11	5.845420e+03
7	8.235430e-03	2.609650e-10	17	8.272403e+12	2.621366e+05
8	1.677722e-01	5.316379e-09	18	3.934641e+14	1.246812e+07
9	3.874205e+00	1.227661e-07	19	1.978420e+16	6.269234e+08
10	1.000000e+02	3.168809e-06	20	1.048576e+18	3.322737e+10

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#)

Closures and Algorithms

[prev](#) | [slides](#) | [next](#)

Algorithms

The time required for this algorithm, when $n=20$, is between 2 and 3 times the age of the universe!

There are, in fact, much more efficient algorithms to form the reflexive transitive closure of R ; one of them is $O(n^3)$.

It's hard to overemphasize how much better this is: if the operation count is indeed n^3 then a problem with $n=400$ will be done in less than one second!

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#)

Closures and Algorithms

[prev](#) | [slides](#) | [next](#)

Algorithms

It is worth noting that knowing the order of an algorithm tells us nothing about how long the algorithm will take. It does tell us something, however, about how the running time will change as the problem size changes.

For example, if an algorithm is $O(n^2)$ we don't know if the operation count is n^2 or $500n^2$ and so we can't make predictions about how long the algorithm will run.

We do know, however, that if the algorithm runs for 5 minutes when $n=100$ then it will run for about 20 minutes when $n=200$. This is because n increased by a factor of 2, so the time should increase by a factor of $2^2=4$.

These estimates are rough, and become more valid as the problem size increases.

[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#)