Constraint Programming

1 year ago

determine the available

determine the available

arrival times

determine the available

arrival times

days

determine the available

arrival times

days

tables

satisfying

satisfying

available tables

satisfying

existing reservations

available tables

satisfying

existing reservations

available tables

opening times

tests

satisfying

existing reservations

available tables

the size of the group

opening times

tests

business rule #1

satisfying

existing reservations

available tables

the size of the group

opening times

tests

business rule #2

business rule #1

satisfying

existing reservations

available tables

the size of the group

release dates

opening times

tests

business rule #2

business rule #3

business rule #1

satisfying

existing reservations

available tables

the size of the group

release dates

opening times

tests
which ones?

business rule #2

business rule #3

business rule #1

satisfying

existing reservations

available tables

the size of the group

release dates

opening times

tests
which ones?

business rule #2

business rule #3

business rule #1

satisfying

existing reservations

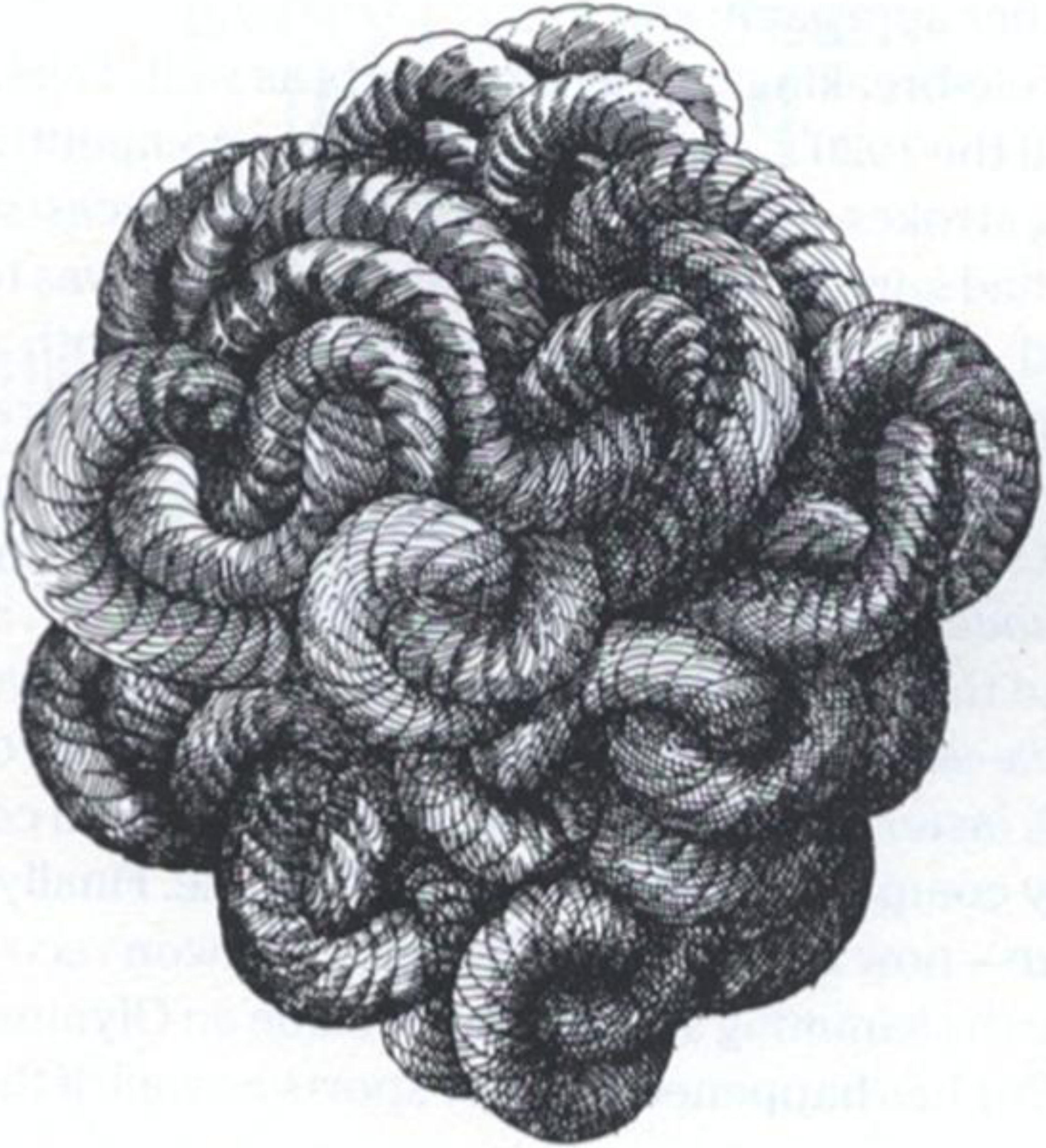
available tables

the size of the group

release dates

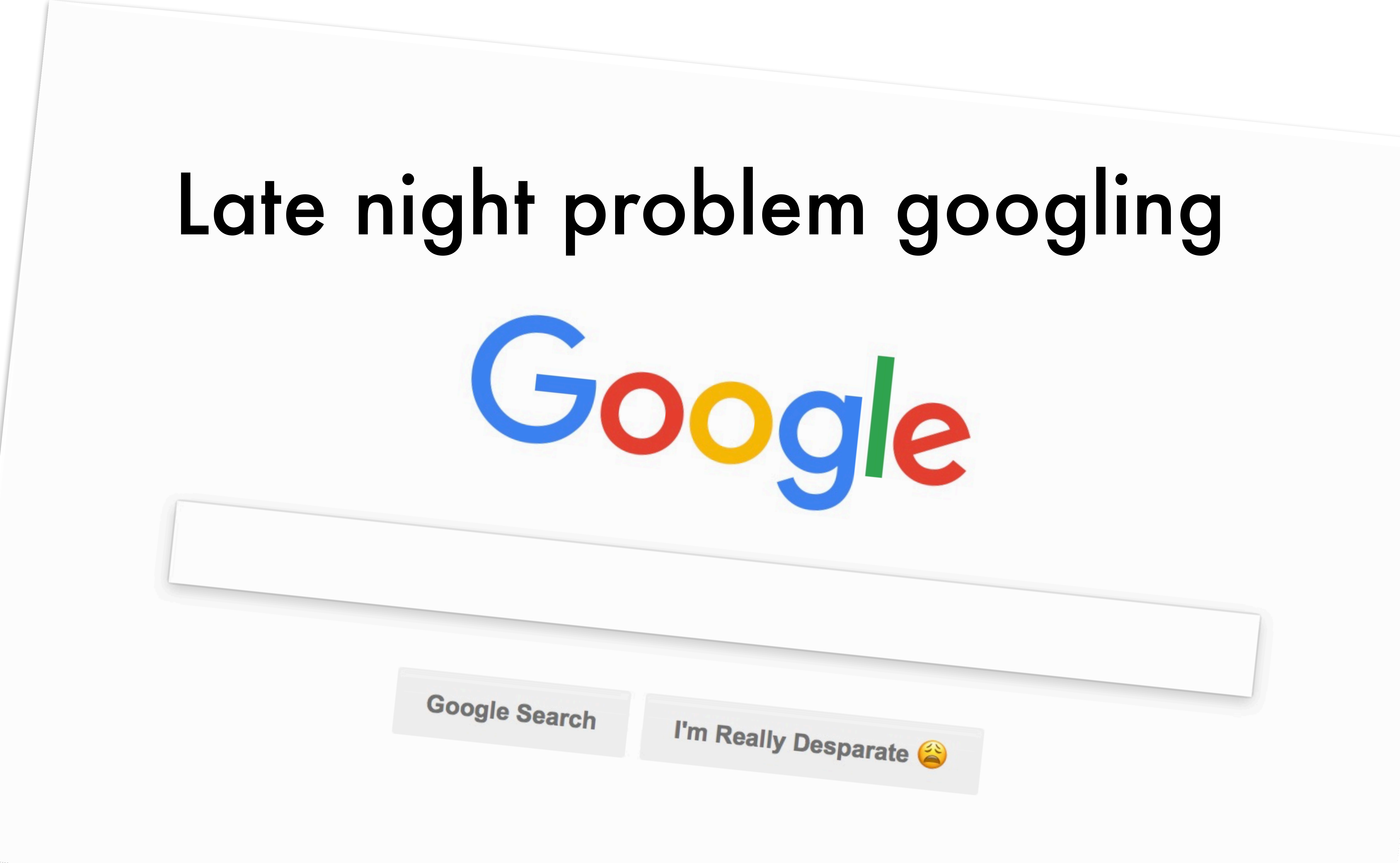
opening times

business rule #137812



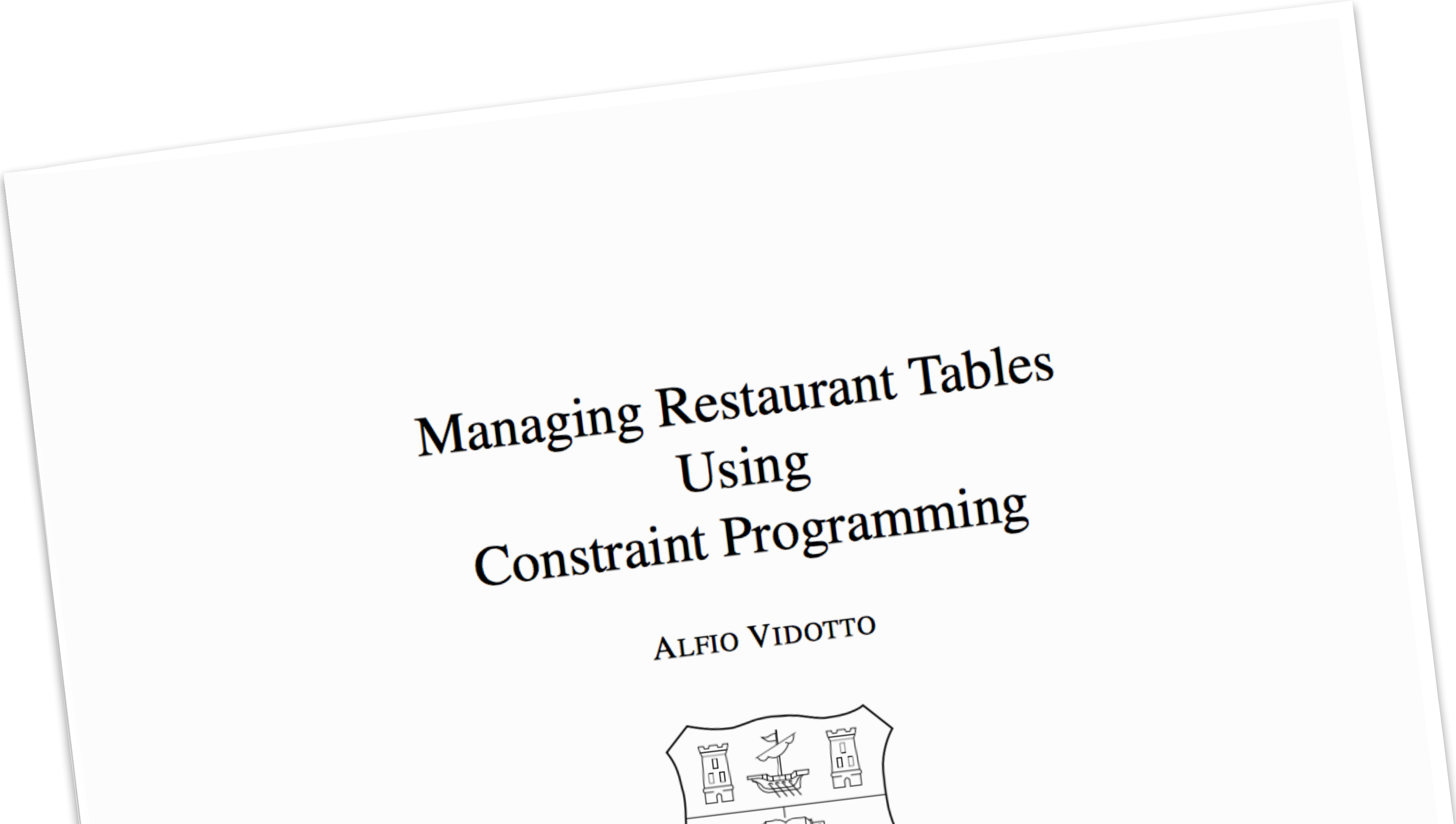
Google
Google Search
I'm Really Desparate 😩

Late night problem googling



‘Managing Restaurant Tables Using Constraint Programming‘

‘Managing Restaurant Tables Using Constraint Programming’



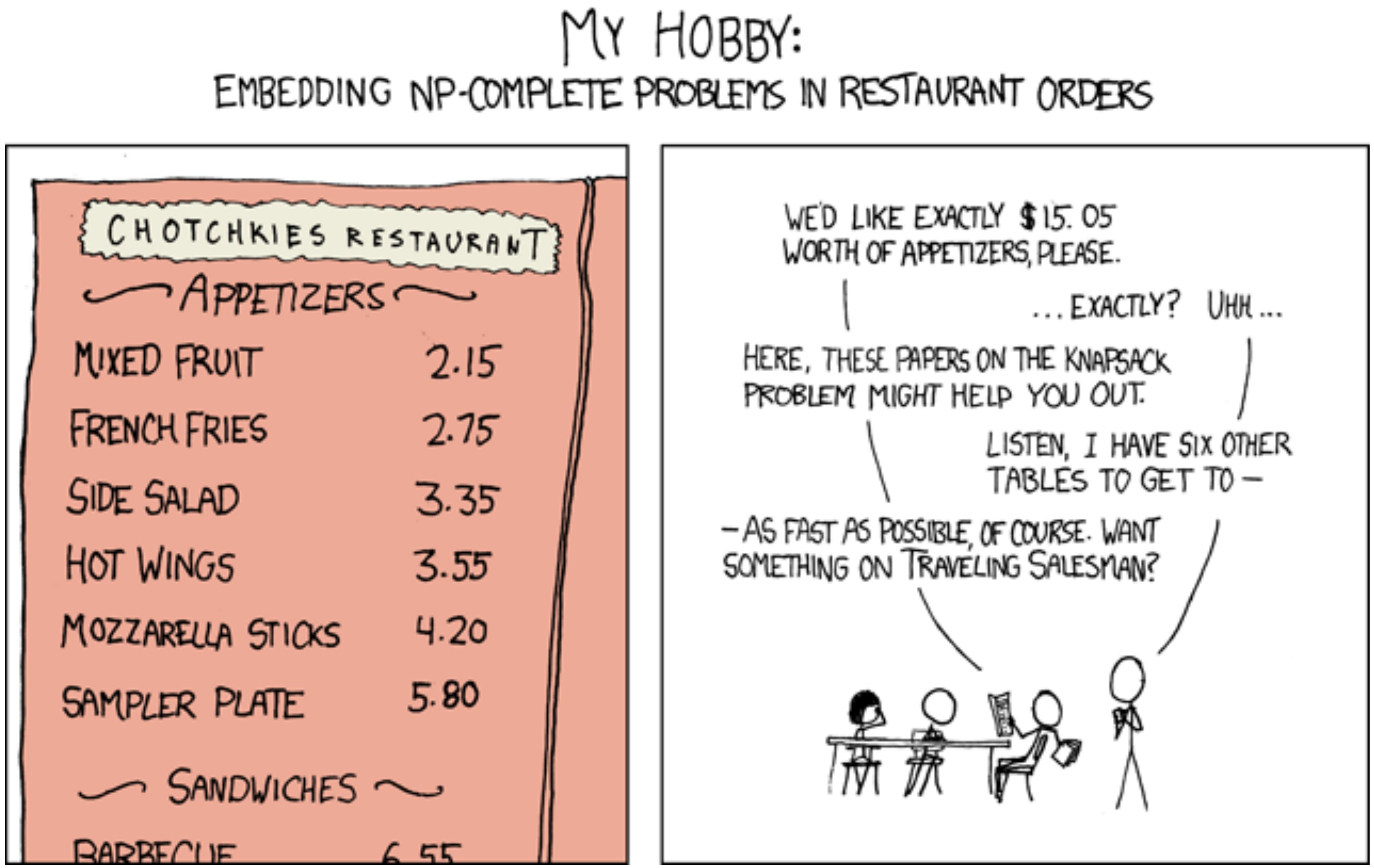
Managing Restaurant Tables
Using
Constraint Programming

Alfio Vidotto

Thank you, Alfio Vidotto

Thank you, Alfio Vidotto

Lets define by example


MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS
CHOTCHKIES RESTAURANT
APPETIZERS
MIXED FRUIT 2.15
FRENCH FRIES 2.75
SIDE SALAD 3.35
HOT WINGS 3.55
MOZZARELLA STICKS 4.20
SAMPLER PLATE 5.80
SANDWICHES
BARBECUE 6.55
WE'D LIKE EXACTLY $15.05 WORTH OF APPETIZERS, PLEASE.
...EXACTLY? UHH...
HERE, THESE PAPERS ON THE KNAPSACK PROBLEM MIGHT HELP YOU OUT.
LISTEN, I HAVE SIX OTHER TABLES TO GET TO –
– AS FAST AS POSSIBLE, OF COURSE. WANT SOMETHING ON TRAVELING SALESMAN?

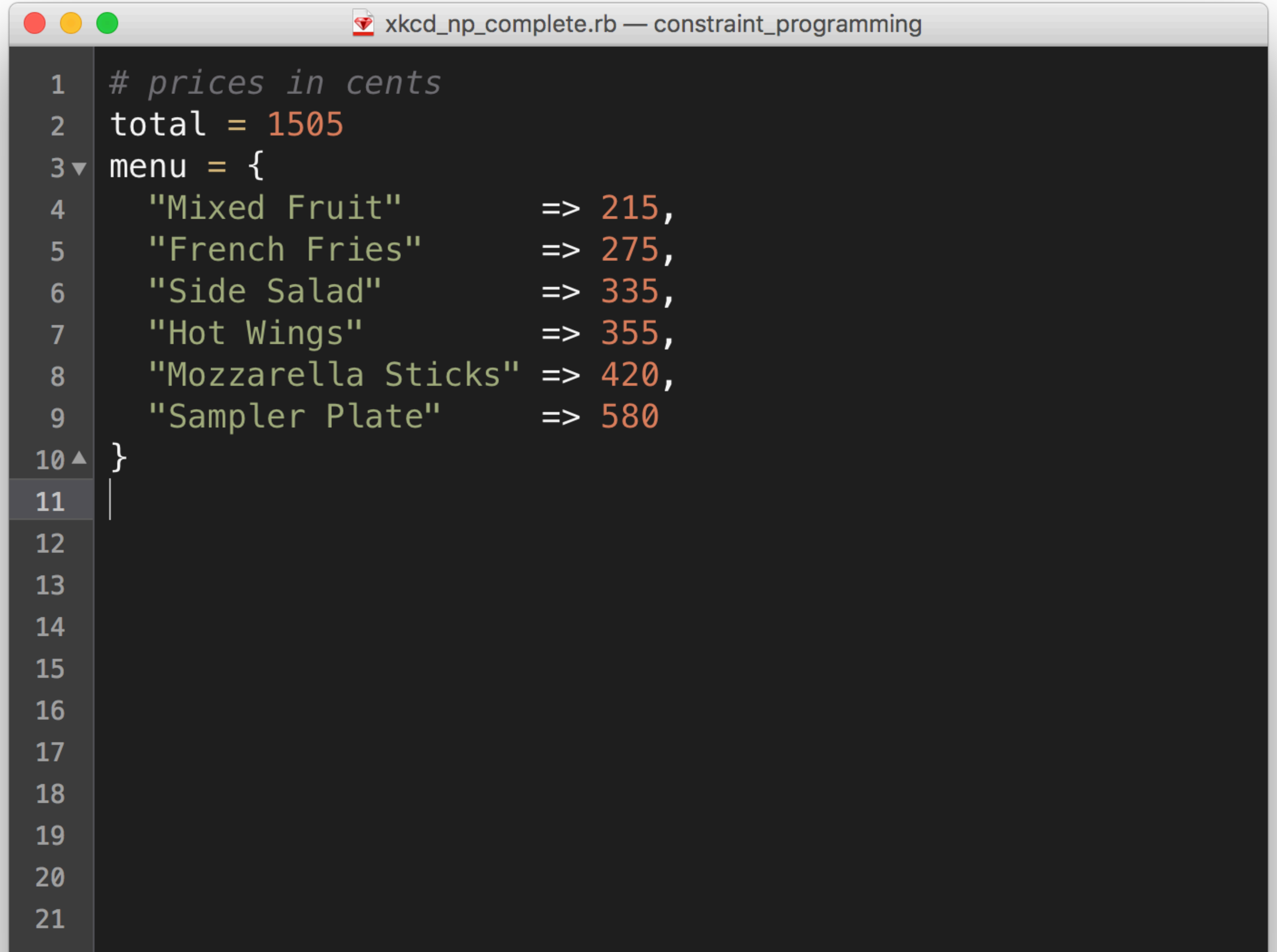
/kən'streɪnt/

A constraint is a logical relation

among several variables,

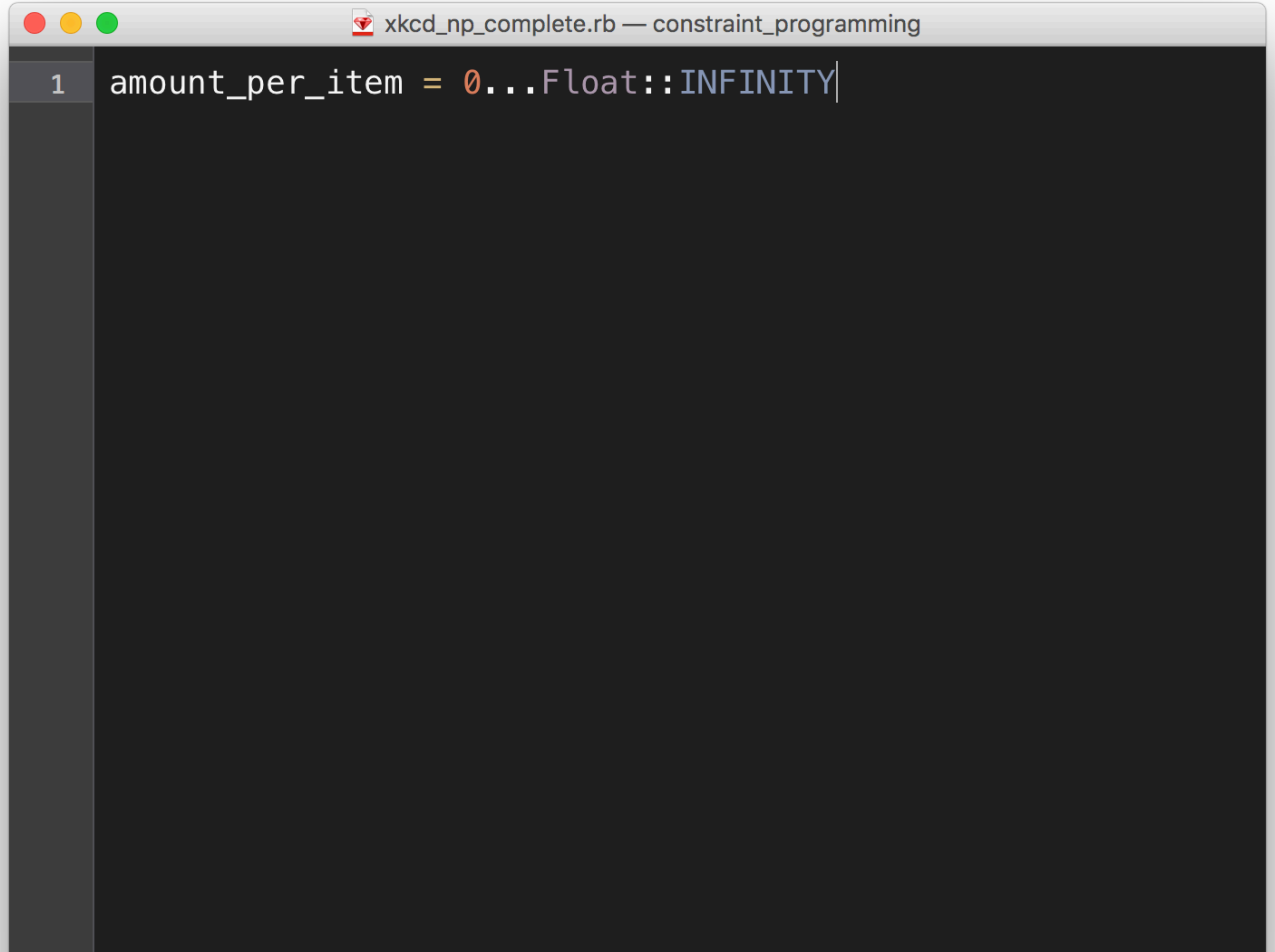
each taking a value in a given domain.

several variables



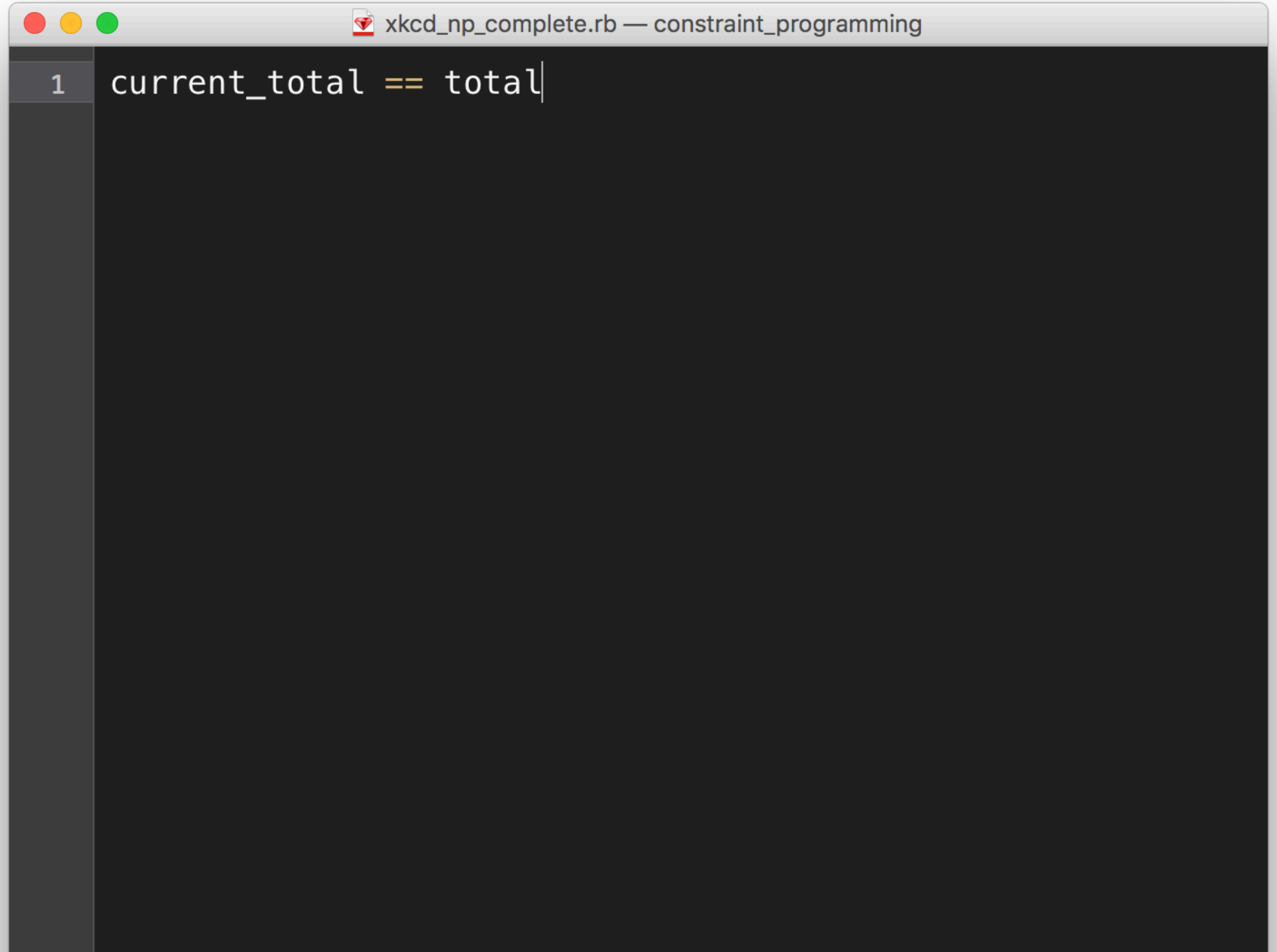
```ruby
# prices in cents
total = 1505
menu = {
  "Mixed Fruit"        => 215,
  "French Fries"       => 275,
  "Side Salad"         => 335,
  "Hot Wings"          => 355,
  "Mozzarella Sticks"  => 420,
  "Sampler Plate"      => 580
}
```

domain



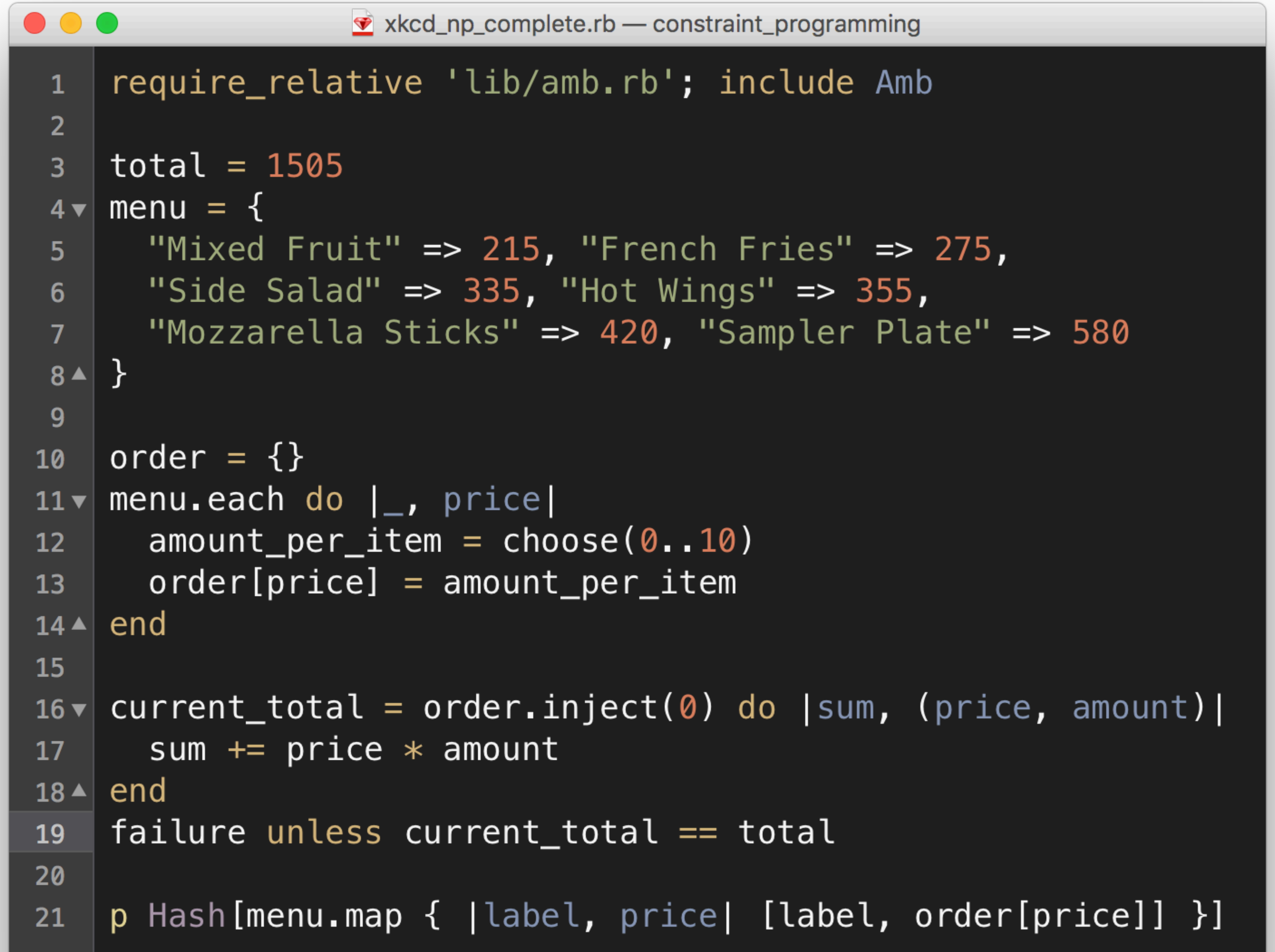
```ruby
amount_per_item = 0...Float::INFINITY
```

logical relation

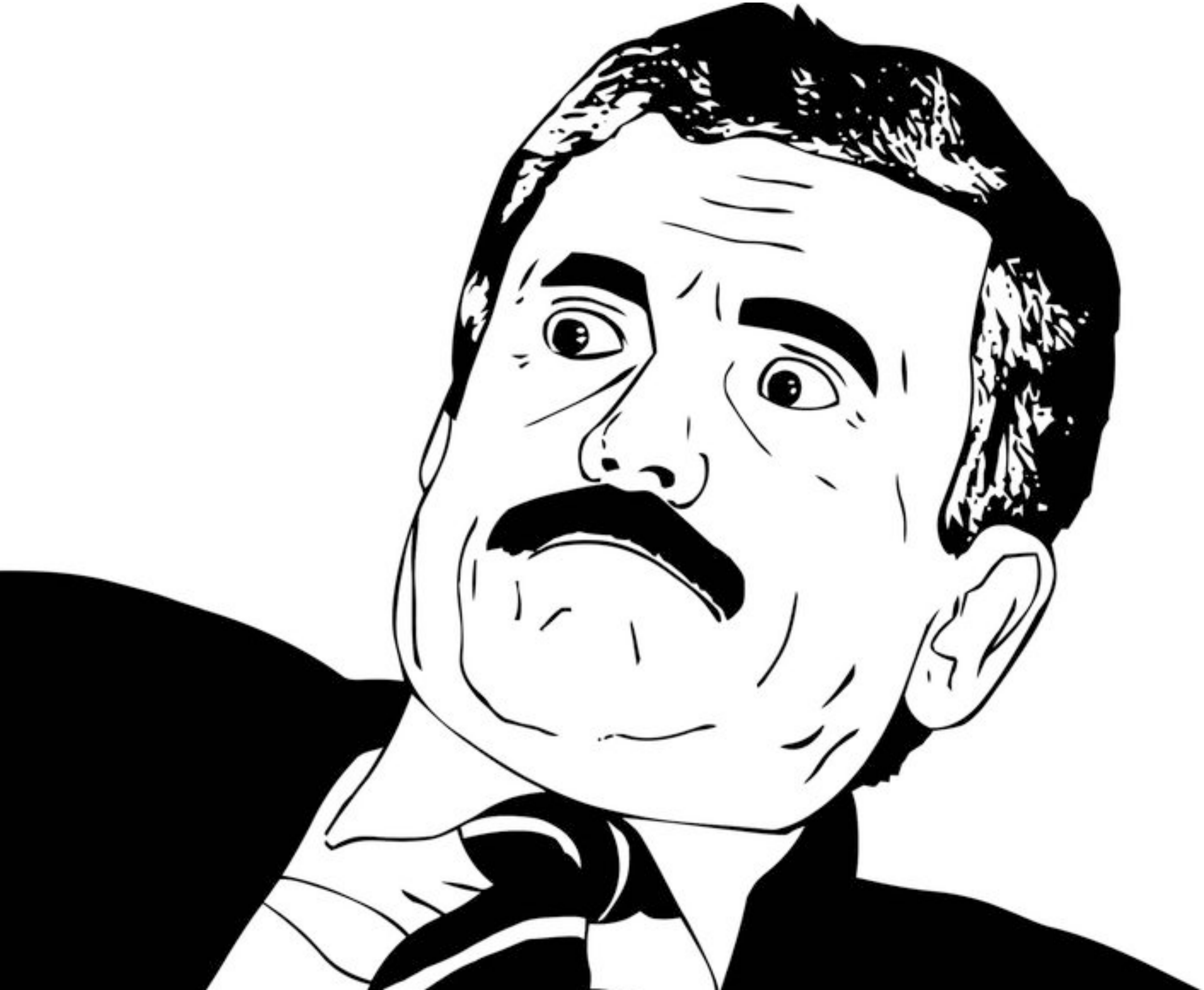


```
current_total == total
```

Easier done than said.



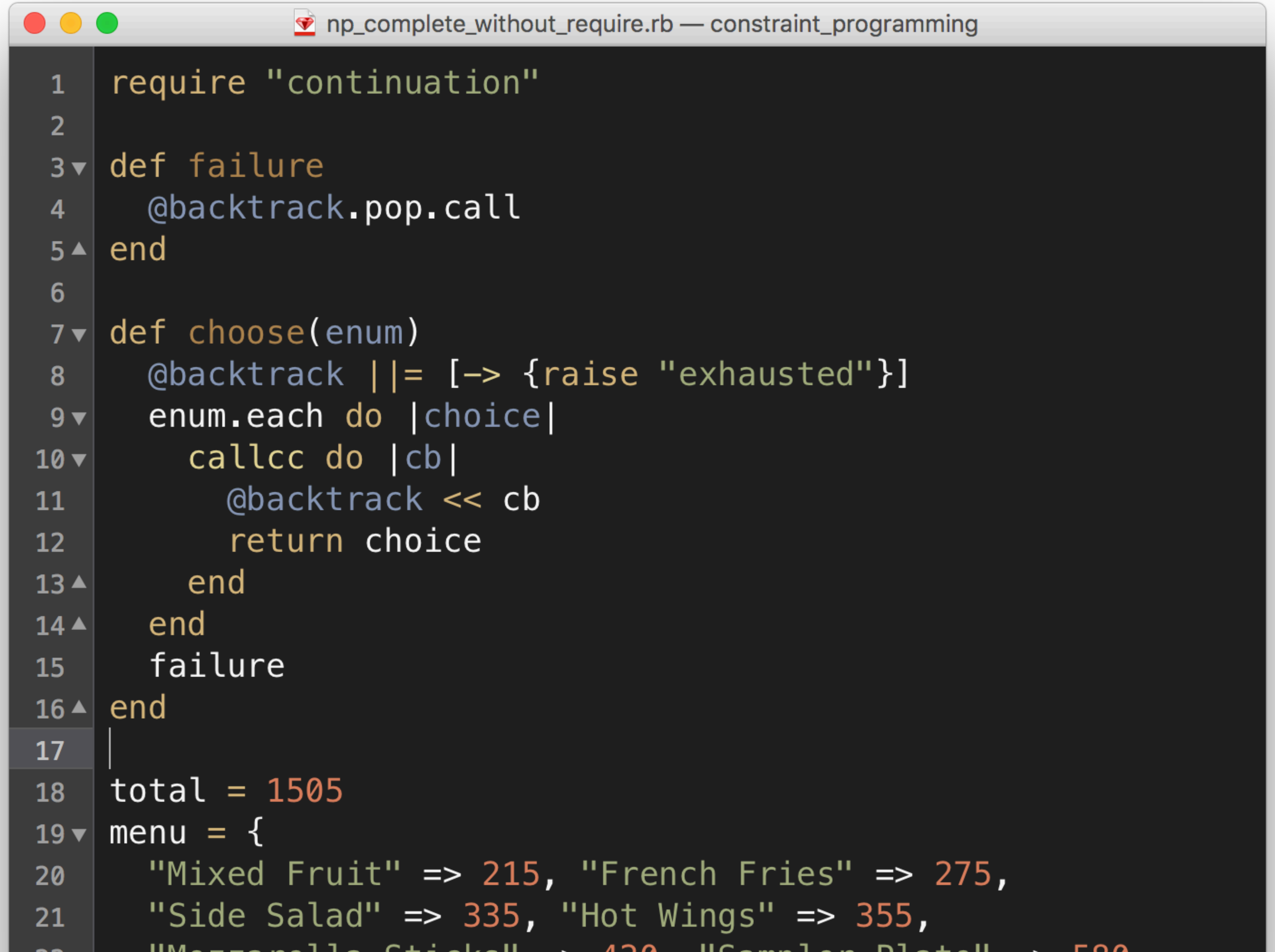
```ruby
require_relative 'lib/amb.rb'; include Amb

total = 1505
menu = {
  "Mixed Fruit" => 215, "French Fries" => 275,
  "Side Salad" => 335, "Hot Wings" => 355,
  "Mozzarella Sticks" => 420, "Sampler Plate" => 580
}

order = {}
menu.each do |_, price|
  amount_per_item = choose(0..10)
  order[price] = amount_per_item
end

current_total = order.inject(0) do |sum, (price, amount)|
  sum += price * amount
end
failure unless current_total == total

p Hash[menu.map { |label, price| [label, order[price]] }]
```



Simple Backtracking in Ruby

meet call/cc

Simple Backtracking in Ruby



```ruby
require "continuation"

def failure
  @backtrack.pop.call
end

def choose(enum)
  @backtrack ||= [-> {raise "exhausted"}]
  enum.each do |choice|
    callcc do |cb|
      @backtrack << cb
      return choice
    end
  end
  failure
end

total = 1505
menu = {
  "Mixed Fruit" => 215, "French Fries" => 275,
  "Side Salad" => 335, "Hot Wings" => 355,
  "Mozzarella Sticks" => 420, "Sampler Plate" => 580
```

Thank you, Jim Weirich

Thank you, Jim Weirich

First research on constraint
satisfaction problems
dates back to the 70s

Montanary 1974
Waltz 1975

Systematic use of constraints
described in

‘Constraint logic programming’
Jaffar, Lassez 1987*

30 years

constraint satisfaction landscape

areas of research			
search	generate and test		
	back tracking	back jumping	back marking
consistency	node consistency	arc consistency	path consistency
constraint propagation	forward checking	look ahead	
reducing search			
constraint optimisation			

use constraint programming

use constraint programming

use constraint programming

for scheduling, planning

use constraint programming

use constraint programming

use constraint programming

for resource allocation*

use constraint programming

Main advantages

Model programs
closely to their real world
entities.

Vast open research
in optimising
difficult problem areas.

Clever modelling can get you around NP-hard problems.

*satisfaction not guaranteed

Interested in more?

Upcoming events

- ACP Summer Schools
  http://www.a4cp.org/events/summer-schools
- CP Conference Series
  http://www.a4cp.org/events/cp-conference-series

  August 28th to September 1st 2017, Melbourne, Australia

Thank you, beloved audience

L.rieder@gmail.com

References

- http://tidel.mie.utoronto.ca/pubs/Theses/vidotto.phd.pdf
- http://rubyquiz.com/quiz70.html
- http://www.math.unipd.it/~frossi/cp-school/CPschool05notes.pdf
- http://kti.mff.cuni.cz/~bartak/constraints/index.html
- http://www.hakank.org/constraint_programming_blog/
- http://www.a4cp.org