Programming Language Support for Relevance

Erik Ernst
Aarhus University, Denmark

WGLD @ Aarhus, 2013

A Conceptual Discussion

- This talk does not present technical results
- Goal: Raising issues at the conceptual level, motivating perspectives, irresponsibly asking for novel semantics

Outline

- Relevance and context
- Context in programming
- Metric spaces delivering context
- Negotiation

Relevance

- Relevance provides complexity reduction: Consider only the relevant phenomena
- We typically ignore it: Pervades everything
- Starting point: The notion of context

Context

- Means environment, surroundings, situation, circumstances, setting — the basis for local interpretation and behavior
- Has been around for billions of years, played a crucial role for all living things
- Makes sense for us!

A built-in feature

- Incomplete knowledge, ambiguity calls for local interpretation: Need context
- Rich representation in animals, including sensory input, hormonal state, memory, experience, communication from others ..
- Used constantly, unconsciously, diversely, subtly or abruptly ..

Language and Context

- Because context is crucial, it emerges in natural language
- More fundamental: Pre-linguistic context dependencies shine through
- Derived, inevitably: Linguistic context works as semantic context as well
- .. incurably intertwined, of course

Syntactic Context

- Compare ‘time flies like an arrow’ and ‘fruit flies like a banana’*
- Several words totally reinterpreted (verb/noun/preposition), accidental sharing in sound and spelling, different parsing, ...
- Not our topic: too many accidental elements

* Not Groucho Marx – see http://en.wikipedia.org/wiki/Time_flies_like_an_arrow;_fruit_flies_like_a_banana

Semantic Context

- Compare

  ‘There was a humongous dog in the book, and the little girl chuckled every time she saw it.’

- This is more manageable

Semantic Context

- Compare

  ‘There was a humongous dog in the room, and the reddish-brown stains on the floor reminded me of the terrible sounds I had heard the previous evening.’

- This is more manageable

Context in Programming

- Traditionally, we think lexical scoping, and maybe inheritance, modules, namespaces
- Almost entirely static
- The dynamic context (object graph) resembles the semantic context
- In general, alignment is a deep challenge!

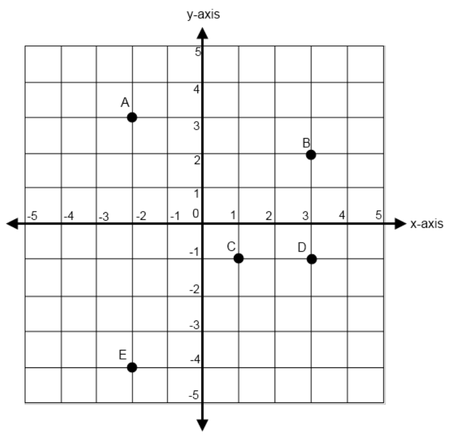
Context in Programming

- Very important case: Transient phenomena, e.g., behaviors, are highly context sensitive, and languages exploit it
- Object-orientation has the context relation method-in-object at its very core
- Class-in-object enables relative transience (e.g., a Ticket may be inside a specific Flight)

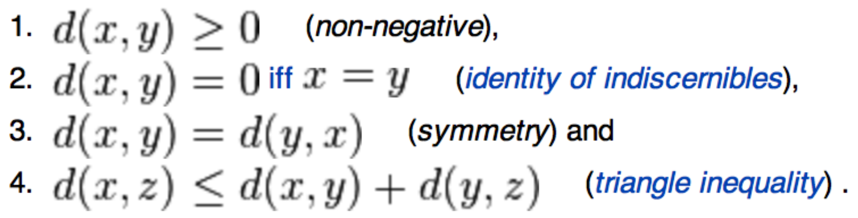
Context in Programming

- How about the dynamics, the object graph?
- Computation touches a few objects at a time: There is a focus area
- Context: reachable objects; objects that can reach “me”, too! No clear boundary!
- Developments in the context are discovered on-demand, not “sensed”

Metric Spaces



- Life generally occurs in metric spaces



1. $d(x,y) \geq 0$ (*non-negative*),
2. $d(x,y) = 0$ iff $x = y$ (*identity of indiscernibles*),
3. $d(x,y) = d(y,x)$ (*symmetry*) and
4. $d(x,z) \leq d(x,y) + d(y,z)$ (*triangle inequality*) .

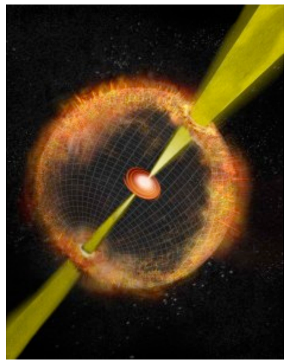
- The mind cannot grasp the entire world at once—and need not
- Immense complexity reduction: Near has full detail, Remote increasingly less
- Works because the world is largely static

Metric Spaces = Relevance

- Metric spaces are organized from “here” and out
- Really good heuristic: Near is Relevant, increasingly remote is increasingly Irrelevant
- Aligning information and importance
- Counter-example Supernova? .. rare

Metric Spaces = Relevance

- Metric spaces are organized from “here” and out
- Really good heuristic: Near is Relevant, increasingly remote is increasingly Irrelevant
- Aligning information and importance
- Counter-example Supernova? .. rare



Understanding the Metric World

- We build understanding on a constantly updated model, where glimpses of information is all we get for updates
- Sensory input is interpreted in the model and used to update the model
- We constantly move, shifting locus and hence *gradually changing* Near/Remote

Metricity in Programming Languages

- Dynamic metricity is tolerable (pointer hops? - and back?!)
- Static metricity is rudimentary (not fading)
- The "world" may not be largely static
- Moving dynamically sort of works, moving statically is traditionally unsupported

Supporting Metricity

- How do we enforce a consistent geometry?
- .. short pointers and long pointers? .. bi-directional? .. enforcing triangle inequality?
- .. protect against surprises “from far away”?
- .. could require name lookups to be at most so-and-so remote, or showing remoteness?

Multiple Simultaneous Contexts

- Easy to envision: Physical context, social network context, online shopping context, etc. It's not about composite contexts.
- This aggravates the danger of conflicts and ambiguities
- My take on this: Avoid it, sequentialize. Don't we always?

Conflict Management

- Multiple contexts is an obvious case for creating conflicts, but not the only one
- Switch of context in real life is often uncoordinated with other activities:
  - On the phone, walk out, drive, ...
- In general, this is hard! E.g., switching behaviors on the stack, respecting invariants

Main Points

- Relevance arises automatically with a context organized as a metric space
- This is extremely old, hence natural!
- Programming languages:
  - Inflexible “on/off” support statically
  - Inconsistent support dynamically
- We need more support for relevance!