Papers We Love 2016

- The Art Of The Propagator
  - Alexey Radul and Gerald Jay Sussman, 2009
- Revised Report on the Propagator Model
  - Alexey Radul and Gerald Jay Sussman

The what now?

- "Place": memory location

The what now?

- "Place": memory location
- "Source": produces the value that is to be put in the place

The what now?

- "Place": memory location
- "Source": produces the value that is to be put in the place
- Most languages: 1 value = 1 source

The what now?

- "Place": memory location
- "Source": produces the value that is to be put in the place
- Most languages: 1 value = 1 source
- But what if we...relaxed that?

Propagator Network

- Places can receive values from multiple sources

Propagator Network

- Places can receive values from multiple sources
- An individual source doesn’t have to know the complete value for a place

Propagator Network

- Places can receive values from multiple sources
- An individual source doesn’t have to know the complete value for a place
- And if places can combine sources, why not delay computations for the resulting value?

Propagator Network

- Places can receive values from multiple sources
- An individual source doesn’t have to know the complete value for a place
- And if places can combine sources, why not delay computations for the resulting value?
- Now we have a "network" of how values flow between places

The simplest thing that could possibly work

- Place = "Cell"

The simplest thing that could possibly work

- Place = "Cell"
  - Stores everything it knows about a single value(or "Nothing")
  - Maintains a set of other propagators to be notified upon change

The simplest thing that could possibly work

- Place = "Cell"
  - Stores everything it knows about a single value(or "Nothing")
  - Maintains a set of other propagators to be notified upon change
- Source = "Propagator"

The simplest thing that could possibly work

- Place = "Cell"
    - Stores everything it knows about a single value(or "Nothing")
    - Maintains a set of other propagators to be notified upon change
- Source = "Propagator"
    - Registers cells that it’s interested in
    - Does something only if it has inputs worth working on

Partial Credit

- A single cell can get information from multiple sources

Partial Credit

- A single cell can get information from multiple sources
- Therefore, the cell acts as the unifier

Partial Credit

- A single cell can get information from multiple sources
- Therefore, the cell acts as the unifier
- Especially when we drop the need for discrete values.

Partial Credit

- A single cell can get information from multiple sources
- Therefore, the cell acts as the unifier
- Especially when we drop the need for discrete values.
- Allow ’merge’ to take intervals!

Barometer abuse

- Make a network that measures the height of a building

Barometer abuse

- Make a network that measures the height of a building
    - Dropping a barometer: h = 1/2gt^2
    - Measuring height of barometer: h = s(hba / sba)

Barometer abuse

- Make a network that measures the height of a building
  - Dropping a barometer: h = 1/2gt^2
  - Measuring height of barometer: h = s(hba / sba)
- Combine!

Barometer abuse

- Make a network that measures the height of a building
  - Dropping a barometer: h = 1/2gt^2
  - Measuring height of barometer: h = s(hba / sba)
- Combine!
- The nontrivial combination of partial information from different sources

So what?

- We can already do that in most languages!

So what?

- We can already do that in most languages!
- Yeah, but since we can merge results, would you believe me if I said we could...

So what?

- We can already do that in most languages!
- Yeah, but since we can merge results, would you believe me if I said we could...
  - trivially add constraints?(stack some mutual inverses inside operators)

So what?

- We can already do that in most languages!
- Yeah, but since we can merge results, would you believe me if I said we could...
    - trivially add constraints?(stack some mutual inverses inside operators)
    - run it in reverse(add a propagator that feeds back into a source cell)

Revised Report interlude

- Cell operations:

Revised Report interlude

- Cell operations:
  - (externally) add content

Revised Report interlude

- Cell operations:
  - (externally) add content
  - collect(merge) accumulated content

Revised Report interlude

- Cell operations:
  - (externally) add content
  - collect(merge) accumulated content
  - register a propagator to be notified

Revised Report interlude

- Cell operations:
  - (externally) add content
  - collect(merge) accumulated content
  - register a propagator to be notified
- Math:
  - Merging must be monotonic, with respect to lattice induced by merge

Revised Report interlude 2

- Propagator operations must be:

Revised Report interlude 2

- Propagator operations must be:
  - commutative

Revised Report interlude 2

- Propagator operations must be:
  - commutative
  - associative

Revised Report interlude 2

- Propagator operations must be:
  - commutative
  - associative
  - idempotent
- (That’s all we need for cells + propagators!)

Dependencies

- if the network observes an inconsistency, we "chuckle" instead of crashing

Dependencies

- if the network observes an inconsistency, we "chuckle" instead of crashing
- This isn’t Star Trek!

Dependencies

- if the network observes an inconsistency, we "chuckle" instead of crashing
- This isn’t Star Trek!
- We can deal with logical inconsistencies just fine, Kirk!

How do we know it’s inconsistent?

- Every piece of data/procedure came from somewhere!

How do we know it’s inconsistent?

- Every piece of data/procedure came from somewhere!
- Either it entered as a premise, or it was created by combining other data.

How do we know it’s inconsistent?

- Every piece of data/procedure came from somewhere!
- Either it entered as a premise, or it was created by combining other data.
- With enough metadata decoration, we can show HOW each datum was derived.

How do we know it’s inconsistent?

- Every piece of data/procedure came from somewhere!
- Either it entered as a premise, or it was created by combining other data.
- With enough metadata decoration, we can show HOW each datum was derived.
    - Yes, this might require more storage space

The World According to Prop

- New definitions: world view

The World According to Prop

- New definitions: world view
- A subset of the data that is supported by a given set of explicit assumptions

The World According to Prop

- New definitions: world view
- A subset of the data that is supported by a given set of explicit assumptions
- IF a contradiction is discovered, the process can now determine WHICH set are "nogood"

The World According to Prop

- New definitions: world view
- A subset of the data that is supported by a given set of explicit assumptions
- IF a contradiction is discovered, the process can now determine WHICH set are "nogood"
- The "chuckle": no computation supported by any superset of those premises can be believed

The truth may be found in one’s pocket

- New definition: Truth Maintenance System

The truth may be found in one’s pocket

- New definition: Truth Maintenance System
    - A system for storing multiple world views

The truth may be found in one’s pocket

- New definition: Truth Maintenance System
    - A system for storing multiple world views
    - A set of items representing direct deductions the surrounding system has added, and any consequences derived.

Truth is complicated!

- When TMSs are merged, the facts are assimilated from the incoming one

Truth is complicated!

- When TMSs are merged, the facts are assimilated from the incoming one
  - Only consequences that are relevant to the current worldview are deduced

Truth is complicated!

- When TMSs are merged, the facts are assimilated from the incoming one
  - Only consequences that are relevant to the current worldview are deduced
  - If we don’t have the consequence yet, feed it back into ourself and check it for consistency

Truth is complicated!

- When TMSs are merged, the facts are assimilated from the incoming one
    - Only consequences that are relevant to the current worldview are deduced
    - If we don’t have the consequence yet, feed it back into ourself and check it for consistency
    - If this results in a contradiction, we yell that no further computations should be done in this worldview

Truth is complicated!

- When TMSs are merged, the facts are assimilated from the incoming one
  - Only consequences that are relevant to the current worldview are deduced
  - If we don’t have the consequence yet, feed it back into ourself and check it for consistency
  - If this results in a contradiction, we yell that no further computations should be done in this worldview
  - If the information in a previous TMS result is already in the new one, we can simply throw it away

Truth is complicated!

- Combining these approaches, we can find the most informative consequence of the current worldview just by using ’merge’!

Never forget

- Remember that the TMS has its own memory!

Never forget

- Remember that the TMS has its own memory!
- Even if we "kick-out" a prior datum, we can bring it back in later on!

Never forget

- Remember that the TMS has its own memory!
- Even if we "kick-out" a prior datum, we can bring it back in later on!
- We also record which ones were bad, so we can throw them out of a search tree.

Never forget

- Remember that the TMS has its own memory!
- Even if we "kick-out" a prior datum, we can bring it back in later on!
- We also record which ones were bad, so we can throw them out of a search tree.
- This ’metadata’ can record HOW we got to a result, not just the result itself

Never forget

- Remember that the TMS has its own memory!
- Even if we "kick-out" a prior datum, we can bring it back in later on!
- We also record which ones were bad, so we can throw them out of a search tree.
- This ’metadata’ can record HOW we got to a result, not just the result itself
- The network will not propagate consequences deducible in an inconsistent worldview

Searching by guessing

- Now allow the receiver to "throw back" results if they are not acceptable

Searching by guessing

- Now allow the receiver to "throw back" results if they are not acceptable
- The sender now knows about this, and must propose an alternate

Searching by guessing

- Now allow the receiver to "throw back" results if they are not acceptable
- The sender now knows about this, and must propose an alternate
    - Now we can allow the network to traverse its own search tree much more efficiently

Searching by guessing

- Now allow the receiver to "throw back" results if they are not acceptable
- The sender now knows about this, and must propose an alternate
    - Now we can allow the network to traverse its own search tree much more efficiently
- This now starts to resemble "dependency-directed backtracking"

Dependency tracking

- If we can move both ways, that could lead to complexity explosion.

Dependency tracking

- If we can move both ways, that could lead to complexity explosion.
- So let’s allow each choice to reverse its decision, and keep track of the ones that worked out

Dependency tracking

- If we can move both ways, that could lead to complexity explosion.
- So let’s allow each choice to reverse its decision, and keep track of the ones that worked out
- If we can do that, why not start making guesses?

Dependency tracking

- If we can move both ways, that could lead to complexity explosion.
- So let’s allow each choice to reverse its decision, and keep track of the ones that worked out
- If we can do that, why not start making guesses?
- With that, we can now manufacture new premises and modify the contradiction detection to inform the guessers of their mistakes(and allow them to change their minds)

Dependency tracking

- If we can move both ways, that could lead to complexity explosion.
- So let’s allow each choice to reverse its decision, and keep track of the ones that worked out
- If we can do that, why not start making guesses?
- With that, we can now manufacture new premises and modify the contradiction detection to inform the guessers of their mistakes(and allow them to change their minds)
- We now have a directed implicit search!

mindblown.gif

- Or more mathy: "a distributed incremental implicit-SAT solver based on propositional resolution"

mindblown.gif

- Or more mathy: "a distributed incremental implicit-SAT solver based on propositional resolution"
- From the point of view of the solver, the problem is implicit in the computation

mindblown.gif

- Or more mathy: "a distributed incremental implicit-SAT solver based on propositional resolution"
- From the point of view of the solver, the problem is implicit in the computation
- From the point of view of the computation, the search done by the solver is implicit

mindblown.gif

- Or more mathy: "a distributed incremental implicit-SAT solver based on propositional resolution"
- From the point of view of the solver, the problem is implicit in the computation
- From the point of view of the computation, the search done by the solver is implicit
- These networks resemble "applicative order lambda calculus": the propagators "push" data through the network.

Tantalizing experiments

Tantalizing experiments

- "The propagator wiring diagram is more analogous to assembly language than expression languages"

Tantalizing experiments

- "The propagator wiring diagram is more analogous to assembly language than expression languages"
    - a virtual machine that implemented ONLY the cell/propagator options?
    - such RPython, very wow

Tantalizing experiments

- "The propagator wiring diagram is more analogous to assembly language than expression languages"
  - a virtual machine that implemented ONLY the cell/propagator options?
  - such RPython, very wow
- "There is no reason to require time to pass uniformly and synchronously in all regions of the network"

Tantalizing experiments

- "The propagator wiring diagram is more analogous to assembly language than expression languages"
  - a virtual machine that implemented ONLY the cell/propagator options?
  - such RPython, very wow
- "There is no reason to require time to pass uniformly and synchronously in all regions of the network"
  - Using the CALM theorem to synchronize cells across remote nodes?
  - Fold it into the "merge" operation?

Revised Report

- The "shop manual", as opposed to the theory

Revised Report

- The "shop manual", as opposed to the theory
- Demonstrates all of the background wiring

Revised Report

- The "shop manual", as opposed to the theory
- Demonstrates all of the background wiring
- Flip between both as necessary

Caveats

- Propagator prototype is written in MIT Scheme

Caveats

- Propagator prototype is written in MIT Scheme
- Also needs "scmutils" package for full performance

Caveats

- Propagator prototype is written in MIT Scheme
- Also needs "scmutils" package for full performance
  - Of course most package managers don’t even include mit-scheme, let alone scmutils

Ramifications and musings

- Compare to the Actor model

Ramifications and musings

- Compare to the Actor model
- Compare to Alan Kay’s vision of what Smalltalk was aiming for

Other people's thoughts

- Lambda the Ultimate poster
    - "this paper is not "we made propagators", it's more about "what if we do all computations through propagators""

Other people’s thoughts

- Lambda the Ultimate poster
    - "this paper is not "we made propagators", it’s more about "what if we do all computations through propagators""
    - "their conclusion might be "you get something similar to a constraint system""

Other people’s thoughts

- A conversation with Sussman

Other people’s thoughts

- A conversation with Sussman
    - "Sussman saw these [asynchronous programming, AI] as interlinked, and that’s what the propagator system is all about! "

Other people’s thoughts

- A conversation with Sussman
  - "Sussman saw these [asynchronous programming, AI] as interlinked, and that’s what the propagator system is all about! "
  - "AI should be "accountable", in the sense that it should be able to express its symbolic reasoning, and be held up to whether or not its assumptions held up to that. "

Credits

- Propagator Network Prototype
  - https://github.com/namin/propagators
- Lambda the Ultimate: The Art of the Propagator
  - http://lambda-the-ultimate.org/node/3250#comment-47997
- A conversation with Sussman
  - http://dustycloud.org/blog/sussman-on-ai

~fin~

- "If you didn’t have fun, we were doing it wrong."