Inflo: Collaborative Reasoning with Open Calculation Graphs

Jonathan Lung
Department of Computer Science
University of Toronto
LungJ@cs.toronto.edu

Steve Easterbrook
Department of Computer Science
University of Toronto
sme@cs.toronto.edu

ABSTRACT
Inflo is a web tool that introduces new ways to collaboratively construct and deconstruct logical arguments drawn as visual dataflow graphs. *Inflo* graphs are dynamic: nodes are logical propositions that can contain computations based on other nodes. *Inflo* nodes and graphs have URLs, permitting sharing via blogs, e-mails, papers, etc. People can collaboratively construct, refine, and adapt/reuse arguments by making changes to shared nodes. By combining community-curated nodes and using *Inflo*'s nodes for computation, back-of-the-envelope calculations can rapidly be made.

Author Keywords
collaboration, knowledge curation, decision making

ACM Classification Keywords
K.4.3 Organizational Impacts: Computer-supported collaborative work

MOTIVATION AND GOAL
Decision making involves evaluating quantitative and/or qualitative factors of each option. Examples include choosing a credit card and picking sustainable food sources. Relevant factors may be hard for an individual to grasp by design (e.g., credit card fees) or innate complexity (e.g., carbon footprints). However, stronger arguments may be built if individuals can contribute sub-arguments from areas of their expertise [4].

Support for complex comparisons is limited. Lists of pros and cons can be used. Back-of-the-envelope calculations might be scrawled on paper or captured in a spreadsheet. Complex calculations may be coded as algorithms where underlying facts and assumptions are hidden. Carbon calculators and university rankings are examples where some or all *factors* are known, but *algorithms and/or inputs* are private.

Visual argument tools like gIBIS and Compendium [1, 2] can handle such arguments, but do not support quantitative analysis or may be too formal for general use. These tools also lack some collaboration features described in Table 1. Open wikis (e.g. Wikipedia) and on-line spreadsheets permit collaboration but are not geared for arguments.

One of this paper's authors compared CO_2 emissions from printing a document versus reading it on-screen using 39 axioms and 15 logical inferences, motivating *Inflo* as a way to show these (along with references) so others could understand, disagree with, and refine the analysis. Our goals were similar to features of contested collective intelligence (CCI) systems identified by De Liddo and Buckingham Shum [3].

Spreadsheets and wikis do not handle knowledge reuse well (see Table 1). In science and maths, new discoveries usually build on previous findings or use previously developed analytical tools. Citations provide *traceability* but not *reuse* because knowledge chunks (such as steps in a calculation or a proof) cannot be *executed* in new contexts. Software-supported decision making may be helped by the equivalent of a programmer's software library. This allows bug fixes (corrections and improvements) in components (logical propositions) to propagate. For example, corrections to our carbon footprint calculation could "patch" copies reused in other contexts, e.g., adapted to cities with different energy mixes or in a comparison of a laptop to iPad.

DESIGN
We built *Inflo*, an HTML5 web app for collaborative reasoning based on argumentation diagrams, to address these problems. Wikipedia is the result of a community collectively curating information; *Inflo* is a body of collectively curated *executable* knowledge. Here we cover *Inflo*'s design and capabilities, a scenario, and future work. *Inflo* has been undergoing small-scale public beta testing since April 2011 by people not involved with its development. Feedback, including from a physicist and a teacher, has prompted design changes and new features. We also hired someone to model parts of Berners-Lee's *How Bad Are Bananas* to identify usability problems. The approach and scenarios sections describe *Inflo* as it exists at the time of writing, unless otherwise specified.

Nodes: Building blocks of arguments
Inflo displays arguments as directed acyclic graphs with each connected subset of nodes drawn as a tree (Fig. 1 depicts a single tree). Each node represents a logical proposition such as "the height of the Empire State Building is 381m tall". Nodes have content that evaluates to a value (e.g., 381 m), a title (e.g., Height of the Empire State Building), and a description field that can be used to explain the content (e.g., providing a hyperlink to the source for the height). In this example, the node's content was a numeric literal, but it

Table 1. Comparing *Inflo*, spreadsheets, and wikis

Functionality	*Inflo*	Spreadsheet	Wiki
Reuse	Individual nodes and graphs can be reused, maintaining history and dependencies. Supports branching.	Templates, copy-paste cell formulae & and contents. May not capture all dependencies.	Templates
Provenance tracing	Individual node author and data source as first class metadata	File-level edit history	Edit history and footnotes
Navigation & storytelling	Follow semantic links, hyperlinks, graph collapsing	Tabular scrolling, hyperlinks	Linear, hyperlinks
Data semantics	Node names, dimensional analysis, implicit conversions	Usually none (e.g., a cell with "50" is meaningless without context)	N/A
Relationship visualization	Forwards, backwards traceability; related info is spatially collocated	Related info may be dispersed within one or over more sheets	N/A
Rationale	Node title, information sources, justifications are first class entities	Cell annotations	Talk page

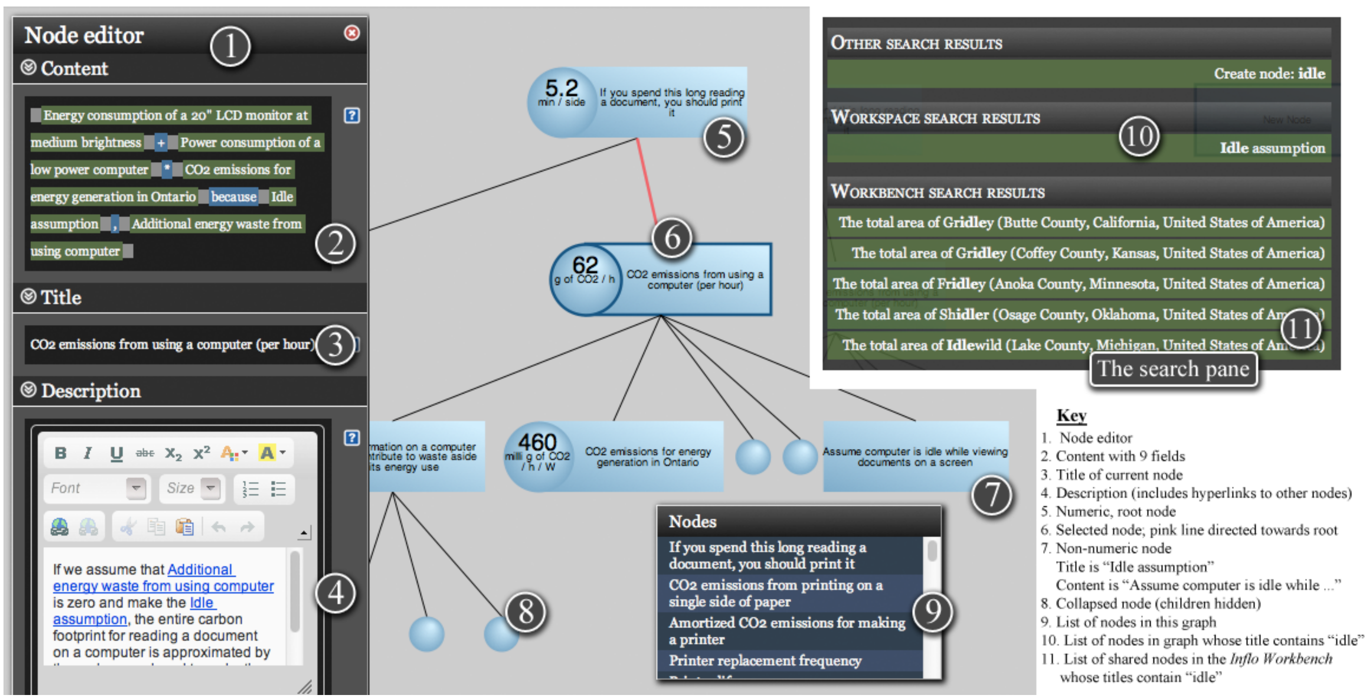

Figure 1. A sample argument shown as an *Inflo* graph and a search window.

could have contained string literals, nodes referenced by title (automatically creating a parent-child relation), and function calls[1]. *Inflo* draws node C as a child of node P iff P depends on/refers to C. Types of references could include because of and contradicts C. Thus, leaves in the tree are axioms; collaboration can be used to dissect these further. Unreferenced and formerly referenced nodes are roots in the forest.

To aid sound reasoning, *Inflo* works with units and alerts users when a dimensional mismatch occurs (e.g., trying to compute 3 m + 2 m/s). Implicit unit conversions may help user comprehension. For example instead of showing the result of 1 N / 9.8 m/s^2 as 1 N*s^2/m, *Inflo* will show 1 kg. Users can use the generic unit "?" to bypass dimensional analysis when using pre-existing formulae that include units. The "?" is propagated towards the root as a reminder that units have not been fully specified.

To avoid burdening users with syntax, few restrictions are placed on node titles (e.g., # of -ve/+ve ions) and units (e.g., 3 C++ books/student). To achieve this, instead of using an automated tokenizer for parsing, users divide a node's content into "fields" using the tab key or mouse (e.g., entering "3 m/s[*tab*]/[*tab*]2 s" results in 3 m/s / 2 s).[2] Any field whose contents are an exact match to a node title or function name are visually colour-coded and treated as node references and function calls, respectively.

Users need not remember titles exactly or the syntax of function calls; an *Inflo*-specific autocomplete widget (Fig. 1 search pane) proposes choices with hints on usage. E.g., typing "arct" shows a dropdown with **arctan** x. When a node

[1] *Inflo* supports user-defined functions written in JavaScript.

[2] Tokenizing is only performed on numeric literals to determine the numeric value and units, using "*" and "/" as delimiters, .

is referenced by title, the tree is redrawn with the referenced node as a child of the referring node. This creates spatial locality for related ideas for navigating and retracing steps during exploration. Nodes referenced more than once are drawn multiple times. Cycles in an argument diagram are sufficient for circular logic, so cycles are forbidden. Thus, *Inflo* arguments are stored as DAGs, but shown as forests.

Example use

The following example was chosen for its simplicity and mix of semantic and numeric values rather than its representativeness of *Inflo* arguments. Consider a theatre selling child tickets for $5 to those under 18 years old and $10 to everyone else. An *Inflo* model could be constructed as follows: First, create a node titled Patron age with content value whose units is "years" (e.g., 16 years). Next, create another node Ticket type with content if Patron age < 18 years then General else Child. The value of Ticket type will evaluate to the string literal General or Child. Lastly, create Ticket price with content if Ticket type = General then 10 dollars else 5 dollars.

Arguments that do not take advantage of *Inflo*'s automated reasoning can also be constructed. Consider three nodes: Mortal premise with content All men are mortal, Socrates' manliness with content Socrates is a man, and Socrates' mortality with content Socrates is a mortal because Mortal premise , Socrates' manliness. *Inflo* will use the supplied conclusion, Socrates is a mortal, as the node's value and automatically draw connections to the referenced nodes. *Inflo* warns users when and where it cannot automatically propagate reasoning, allowing manual updates.

Navigation

Spatial navigation of *Inflo* graphs is similar to the tree/outline view common to file managers. Clicking a node selects it and brings up the node editor (see Fig. 1); a pink line from each visual instance of a selected node is drawn towards the root to show the nodes it influences. *Inflo* supports two methods for managing complex graphs: node collapsing and isolation. Double-clicking nodes collapses them into nondescript circles and hides their children, allowing users to ignore parts of a graph or to emphasize others thus hiding low-level details; this is akin to code-collapsing in an IDE. The node isolation button creates a new workspace with the selected node as the root for focusing on a sub-argument, hiding complexity around and above the node. Users can explore graphs by double-clicking on nodes, panning, scrolling, and zooming.

Versions & URLs: Cornerstones of collaboration

When a node or one of its descendants is changed, a new version is created. Each version of a node has a unique web URL; these URLs are the cornerstones of *Inflo*'s collaborative capabilities. The base URL is of the form http://server/, which points to the *Inflo* web application; this is followed by a URL query string indicating to the web app the unique id of a node and the version to be retrieved. The most obvious advantage to using this scheme is that existing tools that deal with http:// links handle *Inflo* graphs, including blog software, e-mail, and browser bookmarks.

Based on the URL query string, *Inflo* loads the requested node and all its dependencies from a central database. Similarly, a forest of nodes can be retrieved by getting a URL for a particular "view" of a graph. There is no built-in concept of authorization; possessing a node's URL is sufficient for access. Following a link to a graph loads an editable copy. Because a link was provided to specific versions of nodes and edits result in new versions being created, other users following the same link will see the original. Edited graphs can still be shared with others through *Inflo*'s built-in tools or by using conventional link-sharing methods.

The history of a graph or its nodes can be viewed using *Inflo*'s history browser. In a future revision of *Inflo*, users loading an older copy of a graph will be shown an on-screen notification informing them of newer versions of a graph. Likewise, users can also be shown that newer versions of subgraphs exist without affecting the existing view.

Inflo Workbench: Integrated node manager

The *Inflo Workbench* is an *Inflo*-aware collaborative bookmarking system for tracking nodes. Nodes added to the *Inflo Workbench* are not publicly searchable by default, but may be changed on a case-by-case basis. In addition to its collaboration features, *Inflo Workbench* fetches node metadata using node URLs and supports drag-and-drop with *Inflo*.

Opportunistic reuse

There are several ways to reuse nodes within *Inflo*. One way is to drag-and-drop a node's URL onto a graph; sources of the drag-and-drop operation include blog posts, an e-mail links, the *Inflo Workbench*, or a node from *Inflo* running in another window. Since every node has a unique URL, all or parts of a graph can be reused as needed: *Inflo* takes care of node dependencies, so each node can be treated as a discrete, exportable unit. Therefore, users can take advantage of modularity without explicit planning for reuse. Nodes can also be reused by selecting them from the *Inflo*-specific autocomplete widget. This widget searches through functions; nodes in the current workspace; and the user's private collection and all publicly shared nodes in the *Inflo Workbench*. This permits opportunistic reuse of work from others *and* taking advantage of community updates (see below).

Inflo supports basic model merging when conflicts arise. Conflicts occur when a graph contains a different version of a node being imported (either directly or as a dependency). To avoid logical inconsistencies, only one version of a node may exist in a graph at a time. Unused dependencies are removed from the graph rather than forming new trees. Users may choose to convert all instances to either the existing version or the one being imported. *Inflo* never updates versions of nodes without the user's explicit action.

In many or all programming languages and spreadsheets, variable/cell names in a namespace/sheet are unique. *Inflo* node titles need *not* be unique; importing a node with an already-used title does not result in a conflict. No equivalent of variable name shadowing occurs; the autocomplete widget shows all matches with the same title, albeit indistinguishably. Users may opt to drag and drop a node into a field to set the content as desired. However, confusion may still arise. Fortunately, "refactoring" a title is simple; changing the title of a node in the node editor updates all references.

Storytelling

Sometimes, it is necessary to explain why a particular argument or calculation was used. Wikipedia uses wiki (talk) pages for discussions without cluttering the main entry. In addition to providing references, *Inflo*'s description field can serve a similar purpose. Because descriptions can include hyperlinks, including links to other nodes in the graph (which will shift the focus if clicked), it is possible to create narratives about individual nodes and entire graphs (*Inflo*'s help system, another *Inflo* graph, is an example). As software code comments can help explain a codebase, so can the storytelling aspect of *Inflo* be used to help explain a graph.

SCENARIO

Since *Inflo* incorporates the capabilities of both spreadsheets and wikis into one package, it has many applications. A scenario is presented here to illustrate how *Inflo*'s collaborative features create new opportunities and ways for collectively curating knowledge. In this example, a back-of-the-envelope calculation is created, refined, and extended collaboratively.

Back-of-the-envelope calculation

Jono wonders how the carbon footprint of printing compares to reading a document on his low-power desktop computer. With no answer to be found, he begins a calculation in *Inflo*. He first searches *Inflo* for the carbon footprint for printing, but finds nothing suitable. So, he searches the web for information about the manufacture and power consumption of printers. He builds an *Inflo* tree of CO_2 emissions for single-sided printing.

Next, Jono tries to find the carbon footprint of a typical low-power desktop. This information is not yet in *Inflo*, but the local utility company, which shares many nodes with facts about itself such as the number of employees, shares a node with CO_2 emissions per kWh generated. Using that node and a wattmeter, Jono measures his computer's power consumption and derives hourly CO_2 emissions for his computer use.

Jono concludes that reading on his computer at one page per minute produces the same emissions as printing the page. Surprised, he blogs about it. Instead of explaining the entire argument with prose, he drags the root node into his WordPress blog post, creating a link back to his reasoning.

Across the country, Sam, a reader of Jono's blog, is dubious about the analysis. Were CO_2 emissions from paper transport considered? Because of the spatial locality of related concepts in *Inflo* graphs, Sam quickly finds the relevant section, fixes the omission, and clicks "share". Readers of Jono's blog still see Jono's original graph, but also, via an unintrusive popup, that a new version exists.

Elsewhere, Pat, an employee at the local utilities company, updates the CO_2 emissions per kWh node to to reflect lower emissions due to newly-installed wind turbines. Although Pat is unaware of Jono's blog post and Sam's update, people viewing those graphs are notified of the existence of the updated node and have an opportunity to load the latest changes and see the results. Thus, as with software library patches, changes can easily trickle downstream if the API (the node's units) is unchanged.

A year later, Leo wonders about the environmental impact of buying an iPad 3 and cancelling his magazine and newspaper subscriptions. Though unable to find a comparison of the iPad against print media, he does find the latest copy of Jono's analysis via the *Inflo Workbench*. Leo finds the calculation of the CO_2 emissions for paper within Jono's graph; deciding it will be useful for other comparisons, he publicly shares it in the *Inflo Workbench*. He then drags it into his own *Inflo* analysis of the iPad 3, quickly getting an answer to his question.

CONCLUSIONS AND FUTURE WORK

Inflo provides a collaborative environment for organizing information into small, reusable, and traceable chunks. Though the number of user-exposed actions is small – text/calculation entry, spatial & temporal navigation, search, and drag-and-dropping of nodes – it is capable of supporting several different workflows and is generic enough to find uses in a wide variety of domains.

The core collaborative and interactive aspects of *Inflo* have been implemented, but many features are still envisioned. The current implementation of *Inflo* has an architecture designed for extension through the use of plug-ins and APIs allowing third-party developers to build on this work. One of the primary envisioned uses of *Inflo* is as an open carbon calculator; the ability to link to live data sources such as current power consumption has been proposed to support this. Such an addition would be useful in other domains such as business intelligence and logistics.

Because *Inflo* provides a new way of communicating and collaborating, we plan to use it as a platform for exploring how to convey knowledge provenance and trust in decision-making tasks.

ACKNOWLEDGEMENTS

This research was made possible through the generous support of the Business Intelligence Network.

REFERENCES

1. J. Conklin and M. L. Begeman. glbis: A hypertext tool for exploratory policy discussion. *CSCW*, pages 140–152, 1988.
2. J. Conklin, A. Selvin, S. B. Shum, and M. Sierhuis. Facilitated hypertext for collective sensemaking: 15 years on from gibis. *HYPERTEXT*, pages 123–124, 2001.
3. A. D. Liddo and S. B. Shum. Cohere: A prototype for contested collective intelligence. *CSCW Workshop: Collective Intelligence In Organizations*, 2010.
4. A. K. Rantilla and D. V. Budescu. Aggregation of expert opinions. *Hawaii International Conference on System Sciences*, pages 1–11, 1999.