Graph Tracking in Dynamic Probabilistic Programs via Source Transformations

Philipp Gabler,[1] Martin Trapp,[1] Hong Ge,[2] Franz Pernkopf[1]

[1]SPSC Lab, Graz University of Technology, Graz, Austria
[2]Computational and Biological Learning Lab, University of Cambridge, Cambridge, UK

Many machine learning methods require a recorded computation graph of the model they work on. Such graphs can be explicitely defined, or automatically extracted from a model.

We present an approach that can ***automatically record graphs in dynamic models*** (with execution paths varying between runs), while preserving information that is usually only available in static approaches.

We build on ideas from automatic differentiation, where there are two primary implementation strategies [1]:

Operator overloading (OO):

- tracks explicit graph as "Wengert list"
- no knowledge about control structures
- works at runtime, possible overhead
- automatically records graphs dynamically

Source transformation (ST):

- produces new code, no graph structure
- optimizations possible
- can directly inspect control structures
- works at compile time

Our approach combines advantages of OO and ST by transforming code such that it tracks an ***extended Wengert list*** at run time. This structure contains both control and data flow, as well as additional metadata.

The system can be used as tracker in probabilistic programming libraries, such as Turing.jl [2]. It allows to ***inspect control flow***, and trace trough arithmetic operations and function calls.

With the extended Wengert list, it is possible to ***differentiate models*** or implement ***message passing algorithms***. We can also ***analyze the extracted graphs*** to obtain and cache conditionals for efficient Gibbs sampling or automatically ***detect conjugacy*** relationships.

Application Example

$$C \sim \text{Bernoulli}(p) \qquad p \sim \text{Beta}(\alpha, \beta)$$

$$\mu_0 \sim \text{Normal}(M + 2, \sigma_0^2) \qquad M \sim \begin{cases} \text{Normal}(m, s), & \text{if } C = 1 \\ \text{Gamma}(a, b), & \text{otherwise} \end{cases}$$

$$X_i \overset{iid}{\sim} \text{Normal}(\mu_0, \sigma_0^2)$$

Beta is conjugate of *Bernoulli*, so the posterior has closed form. Only if $C = 1$, $M + 2$ has a conjugate Gaussian conditional prior. Note that the value of M will be transformed, and that dynamically tracing trough the addition operator allows us to detect that $M + 2$ is still Gaussian distributed.

The ability to detect and exploit the conjugacy relationships depends on information available only at runtime, and the ability to notice that conjugacy of μ_0 and M is preserved under addition.

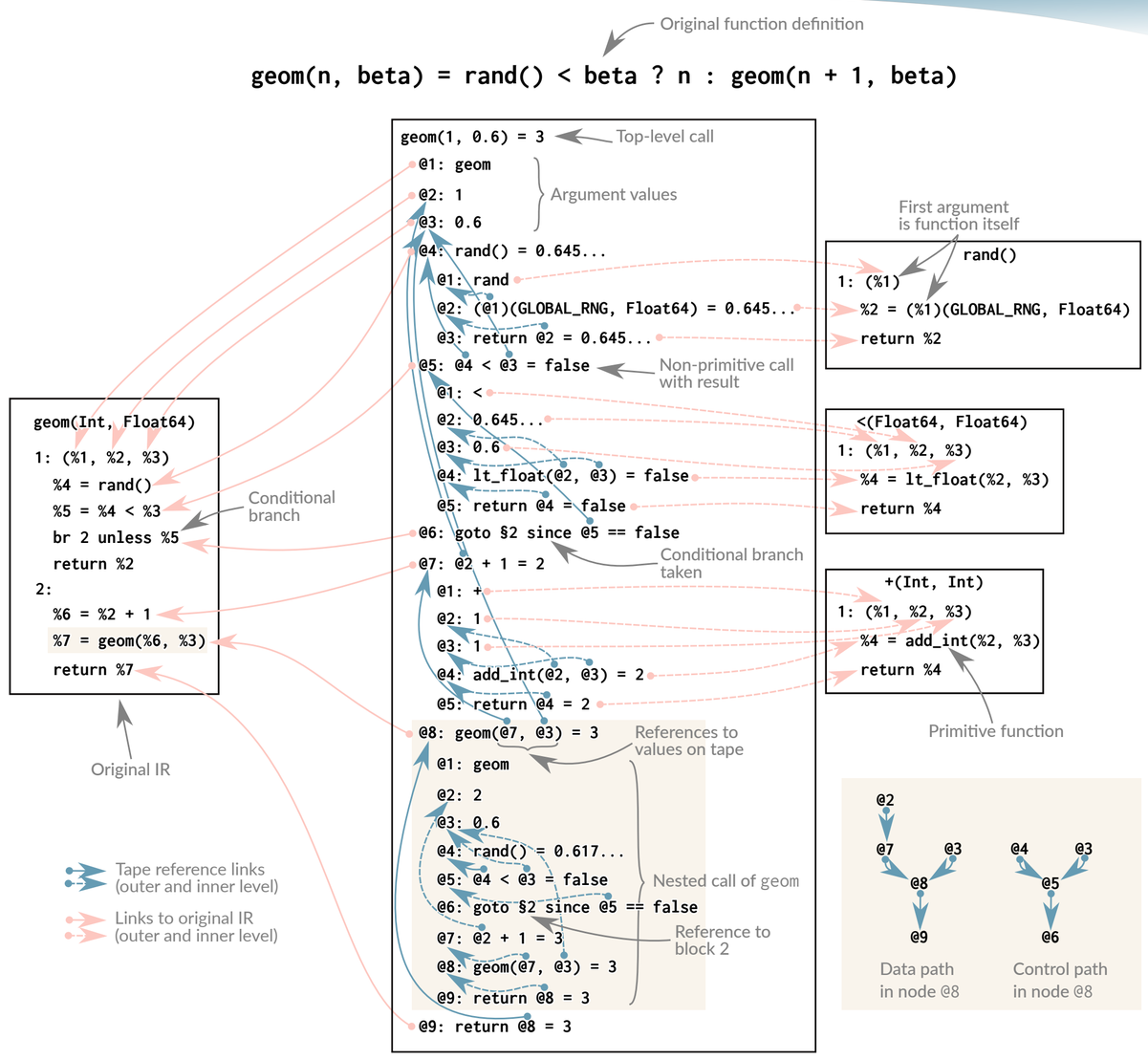
Extended Wengert Lists

To track dynamic computation graphs, we transform the intermediate representation (IR) of the Julia compiler [3]. The input is extended with additional statements to record the operations and jumps executed at runtime as nodes on an extended Wengert list, together with relevant metadata.

A traditional Wengert list is linear, and stores only numerical operations on the data path. Our structure holds sub-lists of all functions called during execution, enriched by recorded control flow decisions and meta information that can be used to analyse the execution.

On the right, we have an example extended Wengert list. It has the following properties:

- two blocks in the original IR
- SSA variables %1 to %7 become tape references @1 to @9
- following back references from @9 results in data path
- non-primitive calls are nested
- intermediate values and jumps are recorded explicitely
- control path is recorded as well, and additional metadata



The geom function draws a sample from the geometric distribution with parameter beta, starting to count at value n. On the left and right, we have the original IR of it and its children.

[1] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, Philadelphia, , 2008.

[2] Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: A Language for Flexible Probabilistic Inference. In *International Conference on Artificial Intelligence and Statistics*, pages 1682–1690.

[3] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1):65–98, 2017. doi: 10.1137/141000671.

2nd Symposium on Advances in Approximate Bayesian Inference
Dezember 8th, 2019
Vancouver, Canada

Find our project on Github:
github.com/phipsgabler/DynamicComputationGraphs.jl

UNIVERSITY OF CAMBRIDGE
TU Graz