VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kansliets noteringar Kod	Dnr
2013-42887-105125-34	2013-4853

2013
Project Grant Junior Researchers

Area of science
Natural and Engineering Sciences
Announced grants
Research grants NT April 11, 2013
Total amount for which applied (kSEK)

2014	2015	2016	2017	2018
929	971	1016	1080	1154

APPLICANT

Name(Last name, First name)
Borgström, Johannes

Date of birth
781229-5553

Gender
Male

Email address
johannes.borgstrom@it.uu.se

Academic title
PhD

Position
Forskarassistent

Phone
0702105435

Doctoral degree awarded (yyyy-mm-dd)
2008-03-10

WORKING ADDRESS

University/corresponding, Department, Section/Unit, Address, etc.
Uppsala universitet
Institutionen för informationsteknologi
Datalogi
Box 337
75105 Uppsala, Sweden

ADMINISTRATING ORGANISATION

Administrating Organisation
Uppsala universitet

DESCRIPTIVE DATA

Project title, Swedish (max 200 char)
Modellkomposition baserad på likelihood i probabilistisk programmering

Project title, English (max 200 char)
Evidence-based Model Composition in Probabilistic Programming

Abstract (max 1500 char)
The field of probabilistic programming aims to make machine learning techniques easier to use and more effective. In this approach, a program serves as a model for the phenomenon of interest. An important property of such a model is the model evidence, a measure of how well the model explains the observed data. However, existing general-purpose languages do not properly support model evidence, limiting the range of models they can express. In this project, we propose to define and study a probabilistic programming language that can make full use of model evidence. We will give the language a compositional semantics, where the meaning of a program can be derived from the meaning of its parts. We plan to study the language from three different aspects: its equational theory, that describes admissible compiler optimizations; its absolute expressiveness, by how well it can describe existing models; and its relative expressiveness, with respect to competing languages. To manage the complexities of measure theory, we will mechanize our results in the theorem prover Isabelle/HOL. To deal with evidence for models of continuous data, we will investigate the use of disintegration: a more general notion of computing probability densities. This work will enable the development of probabilistic programming frameworks that support richer, more accurate and more robust models as programs, to the benefit of AI system developers and data analysts that need to make sense of complex data sets.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-42887-105125-34

Name of Applicant
Borgström, Johannes

Date of birth
781229-5553

Abstract language
English

Keywords
Programming languages, Probabilistic programming, Machine learning, Likelihoods, Machine-checked proofs

Research areas
*Nat-Tek generellt

Review panel
NT-2

Classification codes (SCB) in order of priority
10201

Aspects

Application is also submitted to

similar to:
SSF

identical to:

ANIMAL STUDIES

Animal studies
No animal experiments

OTHER CO-WORKER

Name(Last name, First name)
,

University/corresponding, Department, Section/Unit, Addressetc.

Date of birth

Gender

Academic title

Doctoral degree awarded (yyyy-mm-dd)

Name(Last name, First name)
,

University/corresponding, Department, Section/Unit, Addressetc.

Date of birth

Gender

Academic title

Doctoral degree awarded (yyyy-mm-dd)

Name(Last name, First name)
,

University/corresponding, Department, Section/Unit, Addressetc.

Date of birth

Gender

Academic title

Doctoral degree awarded (yyyy-mm-dd)

Name(Last name, First name)
,

University/corresponding, Department, Section/Unit, Addressetc.

Date of birth

Gender

Academic title

Doctoral degree awarded (yyyy-mm-dd)

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-42887-105125-34

Name of Applicant
Borgström, Johannes

Date of birth
781229-5553

ENCLOSED APPENDICES

A, B, C, N, S

APPLIED FUNDING: THIS APPLICATION

Funding period (planned start and end date)
2014-01-01 -- 2018-12-31

Staff/ salaries (kSEK)

Main applicant	**% of full time in the project**	**2014**	**2015**	**2016**	**2017**	**2018**
Johannes Borgström	40	354	368	383	398	414
Other staff						
Doktorand	80	464	502	532	581	639
Total, salaries (kSEK):		818	870	915	979	1053

Other projectrelated costs (kSek)	**2014**	**2015**	**2016**	**2017**	**2018**
Lokalhyra	46	46	46	46	46
Reskostnader	50	50	50	50	50
Övrigt	15	5	5	5	5
Total, other costs (kSEK):	111	101	101	101	101

Total amount for which applied (kSEK)

2014	2015	2016	2017	2018
929	971	1016	1080	1154

ALL FUNDING

Other VR-projects (granted and applied) by the applicant and co-workers, if applic. (kSEK)

Funds received by the applicant from other funding sources, incl ALF-grant (kSEK)

POPULAR SCIENCE DESCRIPTION

Popularscience heading and description (max 4500 char)
Viktade modeller för komplexa datamängder

Dataexplosionen är ett faktum: idag är det största problemet inte längre att samla in data, utan att utvinna nyttig kunskap från den. Tekniker med namn såsom neurala nätverk, regelbaserade expertsystem och stödvektormaskiner används flitigt inom detta område. En nyare teknik är baserad på probabilistisk programmering, där man helt enkelt skriver ett datorprogram som kan tänkas ha genererat det data man har samlat in, beroende på några okända parametrar. Ett exempel är Microsofts modell för spelarrankning i Xbox Live, där resultatet av en match antas bero på vilken av spelarna som har högst skicklighetsparameter, plus en slumpfaktor. Statistiska tekniker kan sedan användas för att så att säga "köra programmet baklänges" och härleda parametrarna från datat. I Xbox-exemplet kan spelarnas skicklighetsparametrar sedan användas för att ranka spelarna och matcha dem mot jämbördiga motståndare.

I dagsläget är probabilistiska programmeringsspråk betydligt mindre använda och

välförstådda än vanliga språk som Java och Python, och i dagsläget är det det svårt att skriva och använda program som är större än ett tiotal rader. Mitt projekt avser att underlätta skapandet av större välstrukturerade program med hjälp av generella konstruktioner, som modellval eller viktade medelvärden av flera olika modeller för samma data. Sådana "modellblandningar" har fått stor användning inom rekommenderarsystem som t ex hos NetFlix, eftersom den sammansatta modellen representerar antingen konsensus eller ett intelligent val av resultaten av de enskilda enklare modellerna. För att nå absolut säkerhet i resultaten kommer vi att använda oss av automatiska teorembevisare: Datorprogram som kan kontrollera matematiska bevis om de är skrivna i ett speciellt språk.

Ett scenario vi vill stödja är att en utvecklare kan ta ett antal kända modeller och samköra dem mot sitt data, och då får ut en specialanpassad samlingsmodell där den enskilda modell som bäst beskriver en viss del av datat har störst inflytande på motsvarande del av modellen. Detta underlättar förståelsen av komplicerade datamängder, genom att automatiskt reducera dem till mindre delar som var och en har en enkel förklaring.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
þÿ2013-42887-10 12 -

Name of applicant
Borgström, Johannes

Date of birth
þÿ781229- 3

Title of research programme
Evidence-based Model Composition in Probabilistic Programming

Appendix A

Research programme

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Evidence-based Model Composition in Probabilistic Programming

Johannes Borgström

1 Purpose and aims

The rapidly growing field of machine learning has applications both in traditional AI (computer vision, expert systems, etc) and in data analysis, which is an important cross-cutting concern in business, society, and many scientific disciplines (life sciences, physics, economics, sociology and others). *Bayesian machine learning* is a family of techniques that is rapidly growing in popularity in many different fields, due to a strong mathematical foundation and a rich set of possible models and inference techniques. However, non-specialist users either need to use general purpose black-box models and tools, that may not be well-suited for the particular application, or sponsor the development of a customized implementation, which often requires PhD-level skills and significant effort.

One promising approach to increasing the usability and applicability of Bayesian machine learning is *probabilistic programming*, where a model for the system under investigation is implemented as a probabilistic program. Here the definition of the model is separated from training (conditioning the model on the data) and querying (interrogating the trained model for information about the system), allowing for a large degree of separation of labour, and the independent development of the different parts of the system. The view of models-as-programs promises the standard benefits of high-level programming, including encapsulation, abstractions, and software reuse using libraries. To gain maximum benefit, the programming language needs to contain constructs that are suitable for defining and composing probabilistic models in a variety of ways.

An important property of a model is the *model evidence*, that describes how well the observed data is consistent with that model. An important use of model evidence is to compare a family (often parameterised) of models for the same system. It also enables model averaging, where the weight of each model is proportional to its evidence, and other, more complex compositional operators. However, composition operators that make use of model evidence are ill-supported in existing general-purpose probabilistic programming languages, limiting their usefulness for dealing with complex datasets and model uncertainty. The main difficulties arise when considering observed data with a continuous domain (such as the real numbers), which is common in models describing natural phenomena.

In this project, we propose to create a semantic foundation for using model evidence in probabilistic programming languages. The main goal is expressiveness, where we plan to define a language with a set of primitive operations that permit to describe commonly used models and model combinators, that have a well-defined and intuitive meaning (also when dealing with continuous data) and that compose well. In order to cope with the complexities of probability and measure theory, and to avoid the paradoxes and pitfalls of this area, we plan to use the theorem prover Isabelle/HOL to develop a mechanized theory of the language. This will allow

for complete precision in the definitions, and machine-checked proofs for important example programs and facts about the language itself.

The field of probabilistic programming aims to make machine learning techniques easier to use and more effective, by designing and implementing user-friendly modelling languages and their development and run-time environments. Our results will expand the range of models expressible in probabilistic programming languages, enabling scientists, businesses and other non-specialists to develop richer, more expressive and more robust models for their systems of interest and thus facilitating making sense of the data of their domain. A reliable and expressive semantic foundation, as developed in this project, is also an important enabling task for language designers, and is essential in order to exchange models between different systems.

2 Survey of the field

There is a long history of general probabilistic languages with non-termination or failure. These languages were not developed for machine learning purposes, but can still express model evidence in the case of discrete data (only).

Special-purpose probabilistic languages for machine learning either have a semantics that does not permit reasoning about model evidence, or they can only treat finite models for data of deterministic size and admit neither fixpoints nor recursion.

Probabilistic Languages with Non-termination or Failure. Kozen [16] develops probabilistic semantics for while-programs with random assignment. He develops two styles of giving a meaning to programs (operational resp. denotational semantics), and proves them equivalent. McIver and Morgan [17] build on the work of Kozen and develop a theory of abstraction and refinement for probabilistic while programs with real-valued variables, based on weakest preconditions. Their development focuses on demonic nondeterminism in order to describe systems that run in an adversarial context, which is ill-suited for machine learning models. The above languages and several others (including [2, 14] discussed below) allow to encode model evidence for models of discrete data, by failing or entering an infinite loop when the observed data is not equal to the outcome of the program. This approach does not generalize to model evidence for data drawn from continuous distributions, where the probability of a program returning a particular value is identically 0.

Probabilistic Languages for Machine Learning. Several programming models for machine learning mainly represent particular implementation strategies, rather than general Bayesian models. This work includes BUGS [9] and its successors, that use Gibbs sampling for inference on finite Bayesian networks represented as imperative probabilistic programs. An important limitation is the lack of a facility for computing the model evidence for a part of the program.

The perhaps most developed probabilistic programming language framework is Church [10], which represents probability distributions over Lisp terms as Lisp programs (including general recursion), and uses various inference techniques for different classes of models. However, Church only permits conditioning on discrete events (or data). It also does not support computation of model evidence since every admissible Church term denotes a probability distribution (which have model evidence 1 by definition).

Fun is a small (first-order) functional language for the description of Bayesian models [6]. The main novelty of the language is its semantics of programs as *measure transformers*, which

was developed by the PI. This allowed us to add a primitive side-effecting operation for observing data. Fun does support composition operators that rely on model evidence, although observing continuous data is guaranteed to be well-defined only in the special case of an absolutely continuous joint probability distribution for all variables [11].

Infer.NET [18] uses an internal language called Csoft, with a direct semantics as factor graphs using gates [19]. These structures do support calculation of model evidence, but like Fun are only suitable for representing finite models of deterministic size.

Machine-checked probability theory and probabilistic languages. CertiPriv [2] is a library for machine-checked proofs of differential privacy for algorithms expressed in a probabilistic language. However, the current formalization [1] of the mathematical structure (the expectation monad) used to describe the semantics of the language is applicable to discrete data types only. Hurd et al. [14] give a formal semantics to a guarded-command language, combining discrete probabilistic and non-deterministic choice. Here the semantics of a probabilistic program is an expectation transformer, mapping postconditions on final states to weakest preconditions on initial states.

Recent advances in mechanizations of measure theory [13] provide the basic results needed for the semantics of a probabilistic language with continuous variables. We additionally expect to need some standard companion results in the theory of Lebesgue integration, in order to treat mathematical operations such as changes of variable or changes in the order of integration.

3 Project Description

In this project, we propose to develop language primitives and semantics that are sensitive to model evidence in order to implement a variety of standard and custom Bayesian models in a probabilistic programming language. We intend to build on initial work on the language Fun [6], which already has partial support for model evidence (also known as marginal likelihood).

We will initially work descriptively, trying to identify primitive operations that permit to express existing Bayesian models that depend on model evidence. We will seek a *compositional* semantics, where the meaning of a program can be derived from the meaning of its parts. This is essential to support model composition, compositional reasoning about programs and compositional inference, all of which are needed in order to scale to larger models.

We plan to study the resulting semantics from three different aspects: its equational theory, that describes the kinds of refactorings and compiler optimizations that are admissible; its absolute expressiveness, as evidenced by proofs that the semantics of certain program fragments coincides with the mathematical definition of existing models; its relative expressiveness, showing possibilities or impossibilities of encoding results with respect to competing semantics for certain critical models. On an ongoing basis, we will develop formal definitions and machine-checked proofs of important results in the theorem prover Isabelle/HOL, which has a maintainable declarative proof language and existing proof libraries covering important parts of measure theory. To deal with conditioning on continuous data, we intend to investigate the use of *disintegration* [8], which is a more general notion of computing the derivative of a measure.

3.1 Project Components

Survey. At the beginning of the project, we will perform a survey of probabilistic models to collect interesting model composition operators, and evaluate how they can be implemented as functions in a probabilistic language and whether any new primitive operations would be needed. Two examples of such combinators for standard model constructions is an combinator taking the model for a single data point and a connectivity graph into a hidden Markov random field model (e.g., for inferring properties about participants in an online social network given the friendship graph), and a recursive combinator to create partially observed Markov decision processes given each participant's model of the world (e.g., for modelling decision-making under uncertainty). This survey will cover both models that have been implemented in other probabilistic languages, and textbook and research models of Bayesian machine learning.

The survey will take place during the first year of the project.

Probabilistic primitives. The current notion of observation in Fun is only guaranteed to be well-defined for a particular class of models, where all random variables in scope have an absolutely continuous joint distribution. We will attempt to adapt definitions of conditional probability based on disintegration in order to generalise this definition to a larger class of models.

We also intend to treat undirected models, although we envision a directed language. One approach to implementing such models, which can be seen as general graphs, is by stitching together multiple trees (i.e., directed models) using equality constraints. However, these equality constraints are part of the model definition, and need to be implemented using an operation that does not affect the model evidence.

Another interesting operator is nested inference within models, which is for instance used in decision processes where the move of one player depends on what they think their opponent might be doing.

Based on the results of the survey, we will prioritize between these (and possibly other) probabilistic primitives. However, the generalization of observation for continuous variables will be started as early as possible in the project, since sufficient initial evidence exists for its usefulness.

Deterministic primitives. In order to describe various kinds of real-world phenomena such as missing data sequences of unknown length and branching structures such as in phylogenetic trees (the so-called "tree of life"), we will need to add support for recursive models and data of probabilistic size.

Finally, we will need a semantics of higher-order types in order to support object-oriented probabilistic programming, as well as higher-order functions such as maps and folds used in recursive abstractions. Prior work for such limiting processes exists for languages where every statement denotes a (sub-)probability distribution.

Again, these various features will be prioritized in an order dependent on the results of the survey.

Mechanization. We plan to mechanize the work of this project in a theorem-prover, as is now common in the programming languages community. We expect to reap the benefits of full precision in definitions, full confidence in the correctness of proofs, and support in finding the affected parts of proofs whenever the definitions are modified during development. Since the semantics of probabilistic programming languages often make use of non-trivial mathematics,

mechanization of small but crucial litmus examples will be very useful for validation of our definitions. Specifically, we intend to build on recent work in formalization of measure theory in Isabelle/HOL [13].

3.2 Schedule

A rough division of the project into phases is as follows.

Phase 1: Preliminaries

- Mechanization of a continuous probabilistic language with a failure construct, as a baseline language that can later be extended.
- Survey of models and constructions that use model evidence
- Investigation of a more generally applicable definition of observation based on disintegration.

Phase 2: Interlude

- Mechanization of the requisite measure-theory in order to use the disintegration theorem, and applications to some examples.
- Investigation of applicable program transformations (equational theory) for language with failure.
- Collection of litmus examples corresponding to surveyed models; definition of any necessary additional probabilistic primitives.
- Definitions for unbounded models and higher-order program primitives

Phase 3: Language mechanization

- Mechanization of language with disintegration-based observations
- Proofs of litmus examples from surveyed models
- Mechanization of additional primitives and their prerequisites and example models

Phase 4: Follow-through

- Equational reasoning for various primitives, and for the full language as a whole.
- Studies comparing the expressiveness of various (sub-)languages, making use of the language mechanization.
- Examples of larger models, using a combination of new features.

As usual when dealing with mechanizations, especially of mathematics, it is hard to predict the amount of time and effort needed. In case of severe problems, there are still ways of progressing (possibly temporarily): adding standard mathematical results as axioms, mechanization of a weaker primitive operation, or reduction of the scope of the mechanization (i.e., showing only part of the results in a theorem-prover and the rest by hand).

3.3 Outcomes

At the end of the project we expect the following results:

- A good understanding of how to design and use probabilistic languages that treat model evidence for models of continuous data;
- An expressive and maintainable mechanized compositional semantics of probabilistic languages;
- Mechanized proofs that certain programs correctly implement critical model components found in the survey;
- Relative expressiveness results, and possibly translations, relating the proposed language to other proposed probabilistic languages or primitives.
- Extensions of the commons of mechanized mathematics, in particular measure theory, available for any use.

4 Significance

The area of probabilistic programming lies between the fields of machine learning and programming languages. In this project, we investigate how the concept of model evidence from machine learning can be embodied and used in a probabilistic programming language. To this end, we will employ theorem proving techniques and formalized mathematics, as well as ideas from programming language semantics and expressiveness. We hope that the insights from this project will be of direct use in the machine learning community, for instance in addressing the problem of observing deterministically derived variables.

Our model combinators based on model evidence will allow the structured development of large probabilistic models, allowing for the extraction of more refined information about data. As an example, mixtures-of-expert models allow for different models to apply to a lesser or greater extent to different parts of the data, which is a powerful tool for dealing with heterogenous data sets. Large-scale models arising from model compositions using our framework will also in all likelihood stretch or exceed the capabilities of existing inference engines, driving scalability developments in that area.

Our proposed mechanization of a semantics that supports fixpoints, continuous variables and model evidence would allow to mechanize a wide variety of probabilistic languages. For each language, it would enable formal comparisons of expressiveness, studies of language extensions/subsetting, derivation and proof of equational laws, mechanized proofs of the correctness of compilation, and similar foundational work.

In order to exchange probabilistic models between several different frameworks, there must exist an intermediate "smallest common denominator" language where all frameworks can be represented. Also here our mechanized semantics is useful, in order to give the intermediate language a well-defined semantics, and prove correct the translations to and from that language.

We expect the results of this project to appear quickly in the Fun language implementation, and to serve as a guide for developers of other probabilistic languages of how to develop extensions that treat model evidence. The area of probabilistic programming has the aim of making the development of customized machine learning models a routine activity for normal programmers, and is currently set to start growing rapidly. We believe our effort on representing composition operators that have a proven use in machine learning modelling is timely and

likely to gain widespread adoption as part of standard probabilistic programming frameworks to be used by data analysts and AI systems developers worldwide.

5 Preliminary results

Together with collaborators at Microsoft Research in Cambridge I have designed a small (first-order) functional language for the description of Bayesian models [6]. My main contribution to this paper was the *measure transformer* semantics of the language, where a probabilistic program denotes a function that receives an initial joint probability distribution over its free variables, and returns a joint distribution over its variables and result. This formulation of the semantics makes it possible to describe the action of observing a datum as a side-effecting statement, in contrast to existing approaches of encoding observation using an infinite sampling loop [20] or a scoped query construct [10]. We also give a definition of observation of continuous data, which is guaranteed to be well-defined only for absolutely continuous joint probability distributions [11], and show its use in some standard Bayesian models.

Recently [11] we investigate a programming pattern for defining and composing Bayesian models. My main contribution was to show that the semantics of observation allows the implementation of standard composite models such as model averaging [12] and mixtures of experts [15, 5] using a simple if statement, even for continuous data. I also defined a simple combinator for computing model evidence, inspired by a similar construct in factor graphs [19]. Also in this case, the semantics of observation limits the applicability of these combinators and composite models, making its generalisation an important task.

In other recent work [4] we study a number of models with continuous data from the natural sciences. Commonly, a researcher will in addition to their model of the phenomenon under study have to develop and maintain code for computing the probability density of the data returned by the first model. We instead compile the generative model to its density function, avoiding duplication of effort and possible inconsistency. This paper received the EAPLS best paper award at ETAPS 2013.

Taken together[6, 11, 4], we have performed a preliminary investigation for the simpler case of uniformly continuous finite models. The results show a few examples of the usefulness of model evidence, and serve to motivate the proposed work.

My current [7, 21] and ongoing work on the psi-calculi framework and its mechanisation in Isabelle [3] has convinced me of the usefulness of an extensible and maintainable proof archive in programming language research, an experience that the proposed project will greatly benefit from. I also have a strong background in the area of expressiveness of languages; in particular I will be co-chairing the EXPRESS/SOS 2013 and 2014 workshops that treat semantics and expressiveness issues for concurrent programming models.

6 International and national collaboration

Chung-chieh Shan at Harvard has done initial (currently unpublished) work on the use in probabilistic languages of disintegrations for conditional probability distributions, and we plan to collaborate on its extension to model likelihoods. We will collaborate with Andrew D Gordon and Claudio Russo at Microsoft Research Cambridge on the Fun implementation.

6.1 Local collaboration

The mobility group (of which I am a member) has good knowledge in the use and development of Isabelle HOL, especially for formal semantics. I will collaborate with Tjark Weber on Isabelle-specific issues, and Joachim Parrow for language formalization and meta-theory. Other colleagues at the IT department have expertise in programming-language design (Tobias Wrigstad), probabilistic models (Bengt Jonsson and others), and artificial intelligence. In particular, I will collaborate with Michael Ashcroft on issues related to model selection, including priors over models. We will discuss with Svante Jansson at the mathematics department regarding his work on probabilistic graphs and its possible application to modelling data structures of probabilistic size and structure.

7 Independent Line of Research

My work in the field of probabilistic programming started towards the end of my postdoc at Microsoft Research in Cambridge. Since then, I have published a number of papers together with my post-doc advisor and others at MSR. The research at Microsoft focuses more on programmability using probabilistic languages, cf. the Model-learner pattern [11] or compilation of probabilistic programs to density functions [4]. In each of these papers (including [6]) my main technical contributions have been on the semantics and expressiveness of the languages involved. This project proposal lies squarely in the latter line of research.

I am currently the main supervisor for one PhD student, with a more senior researcher as co-advisor, and intend to use the same arrangement for the PhD student financed by this project.

Other grant applications. I currently have a pending application in the area of probabilistic programming, to the SSF future research leader grant; that application focuses on more applied areas such as implementation, heterogenous inference, libraries, and an interchange format. The projects complement each other well but some overlap exists, mainly in the preliminary work and the emphasis on compositionality.

8 Employment Position

As of September 1, 2011, I am a research fellow in the Computing Science division of the IT department of Uppsala university, affiliated with the mobility group. My position is scheduled to end by 31 August 2015. After that point, I plan to move to a researcher position that is partly funded by this project and partly by teaching, if no other suitable position has appeared. However, the start of a machine learning research group was identified as a priority for the department in the most recent peer research assessment of the department, KoF 2011; internal deliberations about department or faculty funding are ongoing.

References

[1] P. Audebaud and C. Paulin-Mohring. Proofs of randomized algorithms in Coq. *Science of Computer Programming*, 74(8):568–589, 2009.

[2] G. Barthe, B. Köpf, F. Olmedo, and S. Zanella Béguelin. Probabilistic relational reasoning for differential privacy. In J. Field and M. Hicks, editors, *POPL*, pages 97–110. ACM, 2012.

[3] J. Bengtson and J. Parrow. Psi-calculi in Isabelle. In *Proceedings of TPHOLs '09*, volume 5674 of *LNCS*, pages 99–114. Springer-Verlag, 2009.

[4] S. Bhat, J. Borgström, A. D. Gordon, and C. V. Russo. Deriving probability density functions from probabilistic functional programs. In N. Piterman and S. A. Smolka, editors, *TACAS*, volume 7795 of *LNCS*, pages 508–522. Springer-Verlag, 2013.

[5] C. M. Bishop and M. Svensén. Bayesian hierarchical mixtures of experts. In C. Meek and U. Kjärulff, editors, *Uncertainty in Artificial Intelligence (UAI'03)*, pages 57–64. Morgan Kaufmann, 2003.

[6] J. Borgström, A. D. Gordon, M. Greenberg, J. Margetson, and J. Van Gael. Measure transformer semantics for Bayesian machine learning. In *European Symposium on Programming (ESOP'11)*, volume 6602 of *LNCS*, pages 77–96. Springer, 2011.

[7] J. Borgström, S. Huang, M. Johansson, P. Raabjerg, B. Victor, J. Åman Pohjola, and J. Parrow. Broadcast psi-calculi with an application to wireless protocols. In G. Barthe, A. Pardo, and G. Schneider, editors, *Software Engineering and Formal Methods: SEFM 2011*, volume 7041 of *LNCS*, pages 74–89. Springer-Verlag, Nov. 2011.

[8] J. T. Chang and D. Pollard. Conditioning as disintegration. *Statistica Neerlandica*, 51(3):287–317, 1997.

[9] W. R. Gilks, A. Thomas, and D. J. Spiegelhalter. A language and program for complex Bayesian modelling. *The Statistician*, 43:169–178, 1994.

[10] N. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: a language for generative models. In *Uncertainty in Artificial Intelligence (UAI'08)*, pages 220–229. AUAI Press, 2008.

[11] A. D. Gordon, M. Aizatulin, J. Borgström, G. Claret, T. Graepel, A. V. Nori, S. K. Rajamani, and C. V. Russo. A model-learner pattern for Bayesian reasoning. In R. Giacobazzi and R. Cousot, editors, *POPL*, pages 403–416. ACM, 2013.

[12] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–401, 1999.

[13] J. Hölzl and A. Heller. Three chapters of measure theory in Isabelle/HOL. In M. C. J. D. van Eekelen, H. Geuvers, J. Schmaltz, and F. Wiedijk, editors, *Interactive Theorem Proving (ITP 2011)*, volume 6898 of *LNCS*, pages 135–151, 2011.

[14] J. Hurd, A. McIver, and C. Morgan. Probabilistic guarded commands mechanized in HOL. *Electr. Notes Theor. Comput. Sci.*, 112:95–111, 2005.

[15] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[16] D. Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22(3):328–350, 1981.

[17] A. McIver and C. Morgan. *Abstraction, refinement and proof for probabilistic systems*. Monographs in computer science. Springer, 2005.

[18] T. Minka, J. Winn, J. Guiver, and A. Kannan. Infer.NET 2.3, Nov. 2009. Software available from `http://research.microsoft.com/infernet`.

[19] T. Minka and J. M. Winn. Gates. In *Advances in Neural Information Processing Systems (NIPS'08)*, pages 1073–1080. MIT Press, 2008.

[20] S. Park, F. Pfenning, and S. Thrun. A probabilistic language based upon sampling functions. In *POPL*, pages 171–182. ACM, 2005.

[21] J. Parrow, J. Borgström, P. Raabjerg, and J. Åman Pohjola. Higher-order psi-calculi, 2012. Accepted for publication in MSCS. Available from `http://www.it.uu.se/research/group/mobility`.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-42887-105125-34

Name of applicant
Borgström, Johannes

Date of birth
781229-5553

Title of research programme
Evidence-based Model Composition in Probabilistic Programming

Appendix B

Curriculum vitae

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Curriculum Vitæ

Johannes Borgström

1 Higher education degree

I hold a Master of Science in Engineering from the Royal Institute of Technology (KTH), Stockholm, awarded on May 9, 2003. My field of studies was *Computer Science and Engineering* and my Master's thesis was on *Translation of smart card applications for formal verification*, supervised by Mads Dam at SICS, Stockholm.

2 Doctoral degree

I received my PhD (*Docteur ès Sciences*) in computer science on the topic of *Equivalences and Calculi for Formal Verification of Cryptographic Protocols* from École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, on March 10, 2008. In my thesis, I developed models and proof techniques for formal verification of cryptographic protocols. My thesis supervisors were Prof. Uwe Nestmann and Prof. Tom Henzinger.

3 Postdoctoral positions

From September 2008 to August 2010 I was a postdoc at Microsoft Research, Cambridge, UK, with Andrew D. Gordon, in the Programming Principles and Tools group headed by Luca Cardelli.

From September 2010 to July 2011 I was a postdoc at the the Uppsala Programming for Multicore Architectures Research Center (UPMARC). I worked in the Mobility group of the IT department at Uppsala University.

4 Docent (n/a)

5 Present position

Since 1 August 2011 I am a Research Fellow (forskarassistent) in the Mobility group of the IT department at Uppsala University. The position is for 80% research, and lasts until July 2015.

6 Previous positions (n/a)

7 Supervision

I am the main supervisor of a PhD student, Ramūnas Gutkovas, starting from August 2011.

8 Deductible time (n/a)

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
þÿ2013-42887-10 12 -

Name of applicant
Borgström, Johannes

Date of birth
þÿ781229- 3

Title of research programme
Evidence-based Model Composition in Probabilistic Programming

Appendix C

List of publications

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Peer-reviewed Scientific Publications

Johannes Borgström

Five main publications are marked with *.

1 Peer-reviewed articles

1. Joachim Parrow, Johannes Borgström, Palle Raabjerg and Johannes Åman Pohjola, *Higher-order Psi calculi*. Accepted for publication in *Mathematical Structures in Computer Science*, 2013. Available from `http://johannes.borgstroem.org/drafts/higher-order-psi.pdf`. *

2. Johannes Borgström, Andrew D. Gordon and Riccardo Pucella, *Roles, Stacks, Histories: A Triple for Hoare*. In *Journal of Functional Programming* **21**(2), Cambridge University Press 2011. An abridged version appeared as chapter 4 of A.W. Roscoe, Cliff B. Jones and Kenneth R. Wood (eds), *Reflections on the Work of C.A.R. Hoare*, History of Computation Series, Springer, 2010.

3. Johannes Borgström and Uwe Nestmann, *On Bisimulations for the Spi-calculus*. In *Mathematical Structures in Computer Science* 15(3), Cambridge University Press, June 2005. *

2 Peer-reviewed conference and workshop publications

4. Johannes Borgström, Ramūnas Gutkovas, Ioana Rodhe and Björn Victor, *A Parametric Tool for Applied Process Calculi*, accepted for *13th International Conference on Application of Concurrency to System Design (ACSD 2013)*.

5. Sooraj Bhat, Johannes Borgström, Andrew D. Gordon and Claudio Russo, *Deriving Probability Density Functions from Probabilistic Functional Programs*, in Proc. *Tools and Algorithms for the Construction and Analysis of Systems*, Rome, March 2013. LNCS 7795, Springer. Awarded the EATCS best paper award at ETAPS 2013. *

6. Andrew D. Gordon, Mihhail Aizatulin, Johannes Borgström, Guillaume Claret, Thore Graepel, Aditya Nori, Sriram Rajamani, and Claudio Russo, *A Model-Learner Pattern for Bayesian Reasoning*, in Proc. *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, Rome, January 2013. ACM Press. *

7. Johannes Borgström, Shuqin Huang, Magnus Johansson, Palle Raabjerg, Björn Victor, Johannes Åman Pohjola, and Joachim Parrow, *Broadcast Psi-calculi with an Application to Wireless Protocols*, in *Proceedings of Software Engineering and Formal Methods*, Montevideo, November 2011. LNCS 7041, Springer.

8. Ioannis Baltopolous, Johannes Borgström and Andrew D. Gordon, *Maintaining database integrity with refinement types*. In *Proceedings of the European Conference on Object-Oriented Programming*, Lancaster, July 2011. LNCS 6813, Springer.

9. Johannes Borgström, Andrew D. Gordon, Michael Greenberg, James Margetson, and Jurgen Van Gael, *Measure Transformer Semantics for Bayesian Machine Learning*. In *Proceedings of the European Symposium on Programming*, Saarbrücken, April 2011. LNCS 6602, Springer. *

10. Johannes Borgström, Juan Chen and Nikhil Swamy, *Verifying Stateful Programs with Substructural State and Hoare Types*. In *Proceedings of PLPV*, Austin, January 2011. ACM Press, 2011.

11. Johannes Borgström, Karthikeyan Bhargavan and Andrew D. Gordon, *A Compositional Theory for STM Haskell*. In *Proceedings of ACM SIGPLAN Haskell Symposium*, Edinburgh, September 2009. ACM Press, 2009.

12. Sven Schneider, Johannes Borgström and Uwe Nestmann, *Towards the Application of Process Calculi in the Domain of Peer-to-Peer Algorithms*. In *Proceedings of Autonomous Systems – Self-Organization, Management, and Control*, Shanghai, October 2008. Springer, 2008.

13. Johannes Borgström, *A Complete Symbolic Bisimilarity for an Extended Spi Calculus*. In *Proceedings of SecCo*, Toronto, August 2008. ENTCS **242**(3), Elsevier.

14. Johannes Borgström, Andrew D. Gordon and Andrew Phillips, *A Chart Semantics for the pi Calculus*. In *Proceedings of EXPRESS*, Lisbon, September 2007. ENTCS **194**(2), Elsevier.

15. Johannes Borgström, Simon Kramer and Olga Grinchtein, *Timed Calculus of Cryptographic Communication*. In *Proceedings of FAST*, Hamilton, Ontario, August 2006. LNCS 4691, Springer.

16. Johannes Borgström, *Static Equivalence* is *Harder than Knowledge*. In *Proceedings of EXPRESS*, San Francisco, August 2005. ENTCS **154**(3), Elsevier.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
þÿ2013-42887-10 12 -
Name of applicant
Borgström, Johannes
Date of birth
þÿ781229- 3

Title of research programme

Appendix N

Budget and research resources

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Appendix N

Evidence-based Model Composition in Probabilistic Programming

The main cost consists of the salary for a PhD. student, working 80% on research over 5 years. The budget includes 40% research time for the PI, and travel costs and office space for the PI and student in proportion to their participation in the project. Miscellaneous costs include general-purpose computer hardware, printing, etc. The remaining 20% of the salary and office space costs for the student are planned to be covered by undergraduate education. For 2014 and 2015, the department finances the reminder of the salary and office space cost for the PI. The indirect costs of the Department of Information Technology are for 2013 calculated as a flat rate of 30% on direct running costs (excl. depreciation of equipment and premises).

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2013-42887-105125-34

Dnr
2013 -

Name of applicant
Borgström, Johannes

Date of birth
781229-5553

Reg date
2013-04-11 19:42:38

Project title
Evidence-based Model Composition in Probabilistic Programming

NT

Research grants NT April 11, 2013
Project Grant Junior Researchers
*Nat-Tek generellt
NT-2
Uppsala universitet

Appendix S

Signatures

2014	2015	2016	2017	2018	
929	971	1016	1080	1154	2014-01-01 -- 2018-12-31

Applicant

Date

Head of department at host University

Clarifi cation of signature

Telephone

A signature on the application is required not only from the applicant but also from the authorised representative of the administrating organisation/institution or equivalent (normally the head of the department or establishment where the research is to be conducted). The signature confirms that the department can accommodate the proposed research, position or equipment; that the costing in the application is approved for the department's part, that any proposed experimentation on human or animal subjects has been reported, and that the applicant has reported any secondary occupations and commercial ties (s)he may have, and nothing inconsistent with good research practice has thereby emerged. The applicant must have discussed these conditions with the representative of the administrating organisation/institution or equivalent before the latter approves and signs the application.

NOTE the exceptions!
Please note that a different procedure applies for the following types of grants:

- Infrastructure- Operation Grants
- Infrastructure- Grants for Expensive Equipment
- Infrastructure- Grants for Large Databases
- Framework Grant SIMSAM

For these types of grants the vice-chancellor, instead of the head of department, must sign Appendix S. Thereby (s)he confirms the commitments of the administrative organization. The details are specified in the Call for Proposals, published at www.vr.se.

Vetenskapsrådets noteringar
Kod

2013-42887-105125-34

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se