Accelerating Healthcare

Improving intelligent mHealth applications
using knowledge compilation

Bachelor Thesis in Artificial Intelligence

Student name:	Mike Overkamp
Student number:	s0640166
Phone number:	+31 6 300 33 965
Email:	m.overkamp@student.ru.nl
External supervisor	Dr. Arjen Hommersom (CS, Department of Model Based System Development)
Internal supervisor:	Dr. ir. Martijn van Otterlo

August 4, 2014

Contents

1 Introduction

In real life situations, many decisions need to be made based on uncertain information. The medical world is no exception to this phenomenon. A medical doctor needs to make the most informed decision based on the information available, for instance a number of symptoms displayed by a patient. Bayesian networks are widely used as a method for dealing with uncertainty [1] and can be used to assist in the decision making process. One area of applications in which these types of networks can potentially fulfill a supporting role, is the field of mobile health (mHealth), which is the active integration of mobile devices in support of general and personal healthcare. Bayesian networks however, do pose a problem in the sense that the computation time increases exponentially with the size of the tree width of the network in the worst case, generally making inference on these networks intractable [2]. This, combined with the fact that computational power on mobile devices is limited, even if it is continuously increasing, means that the real world applications of these networks might be limited. Performing Bayesian network inference on a mobile device may be very slow, if not impossible in some situations. A solution to this problem may lie in a concept known as knowledge compilation, which is the process of compiling some representation of a problem into another in which certain operations are no longer intractable, in an off-line phase, which means that this operation is not performed on the mobile device itself. Those operations which have now become tractable in the new representation can than be performed at a much higher speed in the on-line phase, i.e. on the mobile device itself. Because memory usage, computational power, and the time needed to perform the operations are all decreased, an additional advantage is achieved in the sense that less energy is expended and thus battery time is increased. The difficulty in applying knowledge compilation on a problem is to determine which of many possible target representations is best suited for the problem at hand. In the case of an mHealth application: which target language makes it possible to perform the operations needed in an acceptable time on a mobile device. When the proper target language is determined, the next step is to investigate whether the theoretical improvements of using knowledge compilation are also realized in practice. The research question that I aim to answer in this thesis is therefore multifaceted:

1. *What target compilation language is most suited for mHealth applications that use Bayesian network inference?* and

2. *What are the improvements that can be achieved by utilizing this target language in an existing application?*

The first research question will be answered by analyzing a number of target languages based on their properties and the kinds of operations each of them supports in polynomial time. In order to answer the second research question, the most suited representation will be implemented in an existing mHealth application which is currently being developed at the Radboud University, named eMomCare [3, 4].

In this thesis I will first give a brief overview of the previous research on the subjects of knowledge compilation and mHealth. After that will be a section containing theoretical background information about probability theory, Bayesian networks, weighted model counting and knowledge compilation. The latter section will also contain a separate subsection dedicated to the new compilation language SDD. Following that will be the analysis to determine the best target language. The thesis will conclude with the practical implementation of the chosen target language and a discussion section.

2 Previous research

Knowledge compilation

A lot of research on the subject of knowledge compilation has been done by Darwiche (e.g. [5, 6, 7]). The analysis for the best target language for usage in mHealth projects is largely based on the work done by Darwiche and Marquis [5] as it provides the basis to make an educated selection of candidate languages. Since this paper was published, a lot more research on the subject of knowledge compilation has been done, the most notable being that a new target language SDD [7, 8] has been proposed. As such this language is also taken into account in this thesis. This newer target language is added to the available comparison tables in the section on compilation languages, partly based on previous research by Van den Broeck and Darwiche [9].

mHealth

The World Health Organization defines mHealth as “the use of mobile and wireless technologies to support the achievement of health objectives.” [10] As a result of the substantial increase of the number of available mobile network connections worldwide [11], interest in mHealth applications has increased in recent years as well. mHealth solutions are being used for a large number of different types of healthcare problems across the globe.
A first example in which mHealth applications are utilized is family planning [12]. mHealth applications are also being used as a tool in the fight against HIV and AIDS [13]. In many countries where HIV and AIDS are most prevalent, discussing disease is often taboo. In these countries, mHealth in the form of a simple SMS service offers a great opportunity to reach a large audience for awareness and educational purposes, without sacrificing confidentiality.
More complicated applications offer both physicians as well as their patients a supportive tool to diagnose certain issues [14]. This can either be in the form of a step-by-step guide for the physician or a healthcare worker in order to determine whether or not a certain diagnosis has a high probability, but also give the patient the opportunity to perform regular checkups from their homes. In

the latter case, the physician might offer the patients some tools to take measurements at home after which a mobile application stores the data both locally as well as in the patient's file. An extension of these kinds of applications offers a physician the opportunity to monitor the effectiveness of medication by consulting these measurements, but also the patient's emotional state if it might be altered by prescribed medication. Finally there are those projects that aim to use mHealth applications to monitor and support maternal and child wellbeing during and after pregnancy.

Given the current high costs of health care combined with the increasing need for medical help caused by the aging of society, mHealth applications are a possible method of increasing efficiency and thus reducing health care costs. One of the characteristics that aids towards lower overall costs is reducing the number of face-to-face consultations. It also facilitates a phenomenon referred to as *personalization*, which can be described as the adaptation of general decision making models to a specific patient's individual situation and measurements.

Artificial Intelligence in mHealth

The role of artificial intelligence in mHealth applications can be a very prominent one, depending on the type of application. Of course, in those projects where the mHealth aspect consists of sending a text message make limited use of any artificial intelligence algorithms or theories. This generally changes as the applications become more complicated. Take for instance the mobile heart monitor proposed by Rubel et al. [15], which uses artificial neural networks for the early detection of cardiac events. Minutolo et al. designed an mHealth application for the same purpose using a rule-based decision support system [16]. Finally there are those applications that rely on Bayesian network inference to determine the probability of a certain symptom based on a number of measurements. As reasoning with uncertainty is one of the research field in artificial intelligence and inference is a prominent method for dealing with this type of reasoning, the role of artificial intelligence is apparent. It is this part of the connection between mHealth and artificial intelligence that will be the subject of this thesis.

3 Theoretical background

In order to be able to answer the research questions I aim to answer in this thesis, it is important to first provide a base in the form of the theoretical background on which Bayesian network inference and knowledge compilation are dependent.

Probability theory

Because probability theory is at the basis of probabilistic inference, a short introduction on the subject is given. Probability theory is a means of dealing with uncertainty. In order to explain the background theory used in this thesis three variables are introduced and used as a running example throughout this entire section. Let A, B and C be binary variables, i.e. each can either be true or false. For *instantiations* of these variables the notation a is used to indicate that variable A is true and $\neg a$ and $\overline{a}$ are used interchangeably to indicate that variable A is false. Furthermore $\top$ is the notation for a tautology, i.e. an unconditionally true statement and $\perp$ is the notation for the inverse, an unconditionally false statement. Let $\mathbb{B}$ be a Boolean algebra. The probability distribution P is a function $P : \mathbb{B} \to [0, 1]$, such that $P(\perp) = 0$, $P(\top) = 1$ and $P(x \vee y) = P(x) + P(y)$ if $x \wedge y = \perp$, with $x, y \in \mathbb{B}$. A probability distribution can also be defined over multiple variables. For instance the probability distribution $P(A, B, C)$ is the joint probability distribution over all example variables, where $P(a, b, c)$ represents the probability that all variables are true.

Conditional probability

Conditional probability theory deals with the probability that hypothesis h is true given some evidence e, denoted $P(h \mid e)$. The evidence e consists of all current observations of the world. The conditional probability $P(h \mid e)$ is called the *posterior* probability for h, with $P(h)$, the probability of h without any additional information about any evidence, being the *prior* probability for h. This conditional probability $P(h \mid e)$ can be obtained in the following manner:

$$P(h \mid e) = \frac{P(h \wedge e)}{P(e)}$$

For example, assume we know that if variable A is true, variable B is true with a probability of 0.1, then $P(b \mid a) = 0.1$.

Chain rule

In order to determine the probability for a set of variables $\{X_1, \ldots, X_n\}$, the following rule, which is known as the chain rule, can be utilized:

$$\begin{aligned} P(X_1, X_2, \ldots, X_n) = &P(X_1 \mid X_2, \ldots, X_n)\times \\ &P(X_2 \mid X_3, \ldots, X_n)\times \\ &\vdots \\ &P(X_{n-1} \mid (X_n)\times \\ &P(X_n) \\ = &\prod_{i=1}^{n-1} P(X_i \mid X_{i+1}, \ldots X_n) P(X_n) \end{aligned}$$

When applied to the example variables, this means that:

$$P(A,B,C) = P(A \mid B,C) \times P(B \mid C) \times P(C)$$

Marginalization

In many situations, the probability for a single variable is not part of the known probability distribution. If this is the case this variable can be summed out by utilizing marginalization:

$$\begin{aligned} P(x) &= P(x \wedge \top) \\ &= P(x \wedge (y \vee \neg y)) \\ &= P((x \wedge y) \vee (x \wedge \neg y)) \\ &= P(x \wedge y) + P(x \wedge \neg y) - \text{since } P(a \vee b) = P(a) + P(b) \text{ if } a \wedge b = \bot \end{aligned}$$

Therefore

$$P(x) = \sum_Y P(x,Y)$$

For the example variables, the probability distribution for, for instance, B by itself is not part of the probability distribution. If we want to know the probability for b we simply take the sum of $P(b,a)$ and $P(b,\neg a)$. Using the probabilities as they are given in Figure 1, we obtain the following:

$$\begin{aligned} P(b) &= \sum_A P(b,A) \\ &= P(b \mid a) \cdot P(a) + P(b \mid \neg a) \cdot P(\neg a) \\ &= 0.1 \cdot 0.6 + 0.3 \cdot 0.4 = 0.18 \end{aligned}$$

Bayes' rule

It is often the case that the probability $P(e \mid h)$ is known, but $P(h \mid e)$ is the probability that is most useful. As an example, think of the situation where it is known that some symptom s can be caused by some disease d. The probability that d causes s is known, i.e. $P(s \mid d)$ is known. However, the information that is more interesting in this situation is the probability that a person who suffers from disease d, given that symptom s is present, i.e. $P(d \mid s)$. In order to find the probability when reasoning in the direction opposite of the causal relation, Bayes' rule can be used:

$$P(h \mid e) = \frac{P(e \mid h)P(h)}{P(e)}$$

Let the causal relations for the example variables be as depicted in Figure 1. As can be seen in the Figure A influences both B and C. A could be some disease and B and C two symptoms associated with A. Now say b is observed

$$\begin{tikzcd} & A \arrow[dl] \arrow[dr] & \\ B & & C \end{tikzcd}$$

A	P
a	0.6
$\overline{a}$	0.4

A	B	P
a	b	0.1
a	$\overline{b}$	0.9
$\overline{a}$	b	0.3
$\overline{a}$	$\overline{b}$	0.7

A	C	P
a	c	0.7
a	$\overline{c}$	0.3
$\overline{a}$	c	0.2
$\overline{a}$	$\overline{c}$	0.8

Figure 1: The dependencies and conditional probabilities between the three example variables A, B and C.

and we want to know the probability $P(a)$. Using Bayes' rule we can obtain the probability $P(a \mid b)$:

$$P(a \mid b) = \frac{P(b \mid a)P(a)}{P(b)} = \frac{0.1 \cdot 0.6}{0.18} = 0.33$$

Bayesian Networks

In order to represent conditional relations between a set of variables, *Bayesian networks* can be used. The previously used example in Figure 1 shows the Bayesian network that represents the example variables A, B and C and the conditional relations between these variables. A Bayesian network is formally defined as a pair (G, P), where G is an *acyclic directed graph (ADG)*, with $G = (V, E)$, where V is the collection of vertices and E is the collection of edges. P is the joint probability distribution of a set of variables $\mathbf{X}$, where every variable $X \in \mathbf{X}$ is represented by a node $v_X \in V$ in the network. Every edge $e \in E$ represents a direct relationship between two variables. Two variables X and Y are *independent*, denoted $X \perp\!\!\!\perp Y$, if $P(X \mid Y) = P(X)$ for all instantiations of X and Y. Variables that are not independent may still be *conditionally independent* given some other variables. Two variables X and Y are said to be conditionally independent given variable Z, denoted $X \perp\!\!\!\perp Y \mid Z$, if $P(X \mid Y, Z) = P(X \mid Z)$ for all instantiations of X, Y and Z. In other words: if the value for Z is known, knowing the value for Y does not influence probability for X and vice versa. An example would be that in the example network, once the value for A is known, it no longer matters what the value for C is for the probability that B is true. The conditional (in)dependencies in any Bayesian network imply the conditional (in)dependencies in the probability distribution it represents. By taking these conditional independencies into account, the joint probability distribution P in a Bayesian network can be represented compactly in the form of local conditional probability distributions connected to each node $v \in V$. These are the probability tables at the bottom of the example network.

Bayesian network inference

Bayesian networks are generally used to determine the conditional probability for the value of some variable X in the network given evidence e, for instance to determine $P(a \mid b)$ in the example network. Another possible query to be answered by using a Bayesian network is to determine the maximum a posteriori probability for variable X, or the full probability distribution for X given e, i.e. the value of X that maximizes $P(X \mid e)$, or the distribution $P(X \mid e)$. In order to determine these probabilities, these values need to be inferred from the network. In a simple network this can be achieved by using the marginalization method and the chain rule explained earlier, combined with the exploitation of independencies in the network. However, in many cases this is not a viable option, as it requires the enumeration of all possible combinations of values for all variables in the network.

Variable elimination

One of the most commonly used methods to determine probabilities given a Bayesian network, is an algorithm called *variable elimination*. To explain this algorithm, the first step is to explain *factorization*, which is the process of excluding all variables that are independent of some queried variable from its conditional probability. A factor can be described as a function that transforms a tuple of random variables into a number. An example would be some factor f on the example variables A, B and C, denoted $f(A, B, C)$. $f(a, b, c)$ for instance, is then the numerical value of f if all example variables are true. Because conditional probability distributions can be regarded to be a function over variables and a factor is a representation of such a function, a factor can be used to represent conditional probabilities. For example, the conditional probability $P(B \mid A)$ can be represented as a factor f on A and B, so again assuming both variables are true, $f(b, a) = P(b \mid a)$ would hold. A number of mathematical operations can be performed on factors. The first is multiplication: Let f_1 be a factor over example variables A and B and f_2 a factor over the variables A and C. The product of these two factors, $f_1 \times f_2$, is a factor on the union of the variables:

$$(f_1 \times f_2)(A, B, C) = f_1(A, B) \times f_2(A, C)$$

Secondly, variables can be summed out in a factor: Summing out some variable X from a factor $f(X, Y_0, \ldots, Y_n)$, results in a factor on all remaining variables in the factor $f(Y_0, \ldots Y_n)$. For instance summing out variable A from the previously mentioned factor $f_1(A, B)$, results in a factor on the other variables in f_1, in this case only B:

$$(\sum_A f)(B) = f_1(a, B) + f_1(\neg a, B)$$

Using this operation, the posteriors for any variable X given some evidence can determined by summing the variable out. Of course not all probabilities given some evidence can easily be obtained by summing out a single variable. Often it is needed to sequentially sum out a set of variables The order in which these variables are summed out is an elimination ordering. In the variable elimination algorithm, all variables are summed out given some elimination ordering, until the posteriors for the queried variable are calculated.

Weighted model counting

As said, variable elimination is the most commonly used method to determine some probability in a Bayesian network. However there are alternative methods to find these probabilities. One such alternative is the usage of *weighted model counting (WMC).* WMC is the concept of calculating the weighted sum of all models given a certain theory. A model in this sense is a logical formula that does not contradict the theory. If for example we have observed the example variable A to be true, then $a \wedge b \wedge c$ is a model of this theory, but $\neg a \wedge b \wedge c$ is not. An instance of the WMC is created by defining a logical theory Δ and assigning some weight $W(\ell)$ to each literal ℓ, where a literal is an atomic formula or the negation of an atomic formula, so for instance a and $\neg a$ are literals. These weights then determine the weight for each model ω of Δ as follows:

$$W(\omega) = \prod_{\omega \models \ell} W(\ell)$$

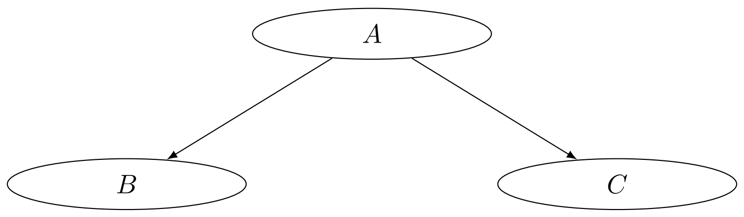
Meaning that it is the product of all literals that are entailed by the model. A literal is entailed by the model it does not contradict the model. The weighted sum $WMC(\Delta)$ is then calculated by:

$$WMC(\Delta) = \sum_{\omega \models \Delta} W(\omega)$$

Which means that it is the sum of all models that satisfy the theory.

Bayesian network to WMC

Any Bayesian network combined with evidence in the form of observations can be transformed into a knowledge base Δ on which WMC can be performed. $WMC(\Delta)$ then corresponds with the probability of the observations. As an example take a Bayesian network with three binary variables $\{A, B, C\}$ as shown in Figure 2, which is identical to the example Bayesian network introduced earlier. In order to perform WMC on the network, the network needs to be transformed into a logical formula. It is generally the case that these formulas are in *negation normal form (NNF)*, which means that the formula consists of only literals and conjunctions and disjunctions. Often a bit more of a restriction is imposed on these formulas and the logical representations are in either *conjunctive normal form (CNF)* or *disjunctive normal form (DNF)*. Formulas



A	P
a	0.6
$\overline{a}$	0.4

A	B	P
a	b	0.1
a	$\overline{b}$	0.9
$\overline{a}$	b	0.3
$\overline{a}$	$\overline{b}$	0.7

A	C	P
a	c	0.7
a	$\overline{c}$	0.3
$\overline{a}$	c	0.2
$\overline{a}$	$\overline{c}$	0.8

Figure 2: A simple Bayesian network

that are in CNF consist of a conjunction of (disjunctions of) literals. A formula in DNF consists of a disjunction of (conjunctions of) literals. To illustrate the how weighted model counting on a Bayesian network works, the example network will be transformed into CNF form. This can be achieved in the following manner.

First the logical variables need to be defined. Each of the possible value for a variable in the Bayesian network needs a corresponding variable in the CNF. This is achieved by defining an indicator variable λ for each possible variable value. Doing this for the network in Figure 2 yields the following logical variables: λ_a and $\lambda_{\overline{a}}$ for the variable A, λ_b and $\lambda_{\overline{b}}$ for the variable B and λ_c and $\lambda_{\overline{c}}$ for the variable C. Logical variables for the conditional probability table entries need to be defined as well. For this, a parameter variable θ is introduced. The following parameter variables are obtained from the example network: θ_a, $\theta_{\overline{a}}$, $\theta_{b|a}$, $\theta_{\overline{b}|a}$, $\theta_{b|\overline{a}}$, $\theta_{\overline{b}|\overline{a}}$, $\theta_{c|a}$, $\theta_{\overline{c}|a}$, $\theta_{c|\overline{a}}$ and $\theta_{\overline{c}|\overline{a}}$.

The next step is to define the knowledge base Δ that represent the Bayesian net. For each instantiation in the network the logical variables whose subscript is consistent with it is set to true. All other variables in the CNF are set to false. The following table shows all instantiations for the network in Figure 2 and the corresponding CNF variables that are set to true (all other variables in the CNF are set to false).

Network instantiation	CNF variables set to true
abc	$\lambda_a\lambda_b\lambda_c\theta_a\theta_{b\|a}\theta_{c\|a}$
$ab\overline{c}$	$\lambda_a\lambda_b\lambda_{\overline{c}}\theta_a\theta_{b\|a}\theta_{\overline{c}\|a}$
$a\overline{b}c$	$\lambda_a\lambda_{\overline{b}}\lambda_c\theta_a\theta_{\overline{b}\|a}\theta_{c\|a}$
$a\overline{b}\overline{c}$	$\lambda_a\lambda_{\overline{b}}\lambda_{\overline{c}}\theta_a\theta_{\overline{b}\|a}\theta_{\overline{c}\|a}$
$\overline{a}bc$	$\lambda_{\overline{a}}\lambda_b\lambda_c\theta_{\overline{a}}\theta_{b\|\overline{a}}\theta_{c\|\overline{a}}$
$\overline{a}b\overline{c}$	$\lambda_{\overline{a}}\lambda_b\lambda_{\overline{c}}\theta_{\overline{a}}\theta_{b\|\overline{a}}\theta_{\overline{c}\|\overline{a}}$
$\overline{a}\overline{b}c$	$\lambda_{\overline{a}}\lambda_{\overline{b}}\lambda_c\theta_{\overline{a}}\theta_{\overline{b}\|\overline{a}}\theta_{c\|\overline{a}}$
$\overline{a}\overline{b}\overline{c}$	$\lambda_{\overline{a}}\lambda_{\overline{b}}\lambda_{\overline{c}}\theta_{\overline{a}}\theta_{\overline{b}\|\overline{a}}\theta_{\overline{c}\|\overline{a}}$

This strategy works for the small example network used, however listing all possible instantiations of a real world network will generally be intractable. A more efficient manner of representing the knowledge base as a CNF is by processing each network variable and each parameter. Below is an example of a CNF encoding for the example network. In the table CPT stands for conditional probability table. More background information on CNF encoding can be found in Chavira and Darwiche [6].

Variable A	$\lambda_a \vee \lambda_{\overline{a}}$	$\neg\lambda_a \vee \neg\lambda_{\overline{a}}$
Variable B	$\lambda_b \vee \lambda_{\overline{b}}$	$\neg\lambda_b \vee \neg\lambda_{\overline{c}}$
Variable C	$\lambda_c \vee \lambda_{\overline{c}}$	$\neg\lambda_c \vee \neg\lambda_{\overline{c}}$
CPT 1	$\lambda_a \Leftrightarrow \theta_a$	$\lambda_{\overline{a}} \Leftrightarrow \theta_{\overline{a}}$
CPT 2	$\lambda_a \wedge \lambda_b \Leftrightarrow \theta_{b\|a}$	$\lambda_{\overline{a}} \wedge \lambda_b \Leftrightarrow \theta_{b\|\overline{a}}$
	$\lambda_a \wedge \lambda_{\overline{b}} \Leftrightarrow \theta_{\overline{b}\|a}$	$\lambda_{\overline{a}} \wedge \lambda_{\overline{b}} \Leftrightarrow \theta_{\overline{b}\|\overline{a}}$
CPT 3	$\lambda_a \wedge \lambda_c \Leftrightarrow \theta_{c\|a}$	$\lambda_{\overline{a}} \wedge \lambda_c \Leftrightarrow \theta_{c\|\overline{a}}$
	$\lambda_a \wedge \lambda_{\overline{c}} \Leftrightarrow \theta_{\overline{c}\|a}$	$\lambda_{\overline{a}} \wedge \lambda_{\overline{c}} \Leftrightarrow \theta_{\overline{c}\|\overline{a}}$

The following step is to assign a weight $W(\ell)$ to each literal in the CNF. Each positive literal of a parameter variable is assigned a weight equal to the probability of the corresponding conditional. All other literals are assigned a weight of 1. In the example, this means that all weights are 1 except the following:

$$\begin{array}{ll} W(\theta_a) = 0.6 & W(\theta_{\overline{a}}) = 0.4 \\ W(\theta_{b|a}) = 0.1 & W(\theta_{\overline{b}|a}) = 0.9 \\ W(\theta_{b|\overline{a}}) = 0.3 & W(\theta_{\overline{b}|\overline{a}} = 0.7 \\ W(\theta_{c|a}) = 0.7 & W(\theta_{\overline{c}|a}) = 0.3 \\ W(\theta_{c|\overline{a}}) = 0.2 & W(\theta_{\overline{c}|\overline{a}}) = 0.8 \end{array}$$

Because these are the only weights that are not equal to 1, each model ω has a weight that is equal to the product of the weights of all positive literals.

The final step in transforming Bayesian network inference into an instance of weighted model counting is to add observations. Adding evidence can be achieved in two different ways. The first is to change the weights of all indicator variables λ whose subscript contradicts the evidence from 1 to 0. The result is that the weights for all models that not support the evidence is now zero. The following table illustrates this for the example with added evidence $e = \{a, b\}$.

Network instantiation	CNF variables set to true	Weight without e	Weight with e
abc	$\lambda_a\lambda_b\lambda_c\theta_a\theta_{b\|a}\theta_{c\|a}$	$0.6 \cdot 0.1 \cdot 0.7 = 0.042$	0.042
$ab\overline{c}$	$\lambda_a\lambda_b\lambda_{\overline{c}}\theta_a\theta_{b\|a}\theta_{\overline{c}\|a}$	$0.6 \cdot 0.1 \cdot 0.3 = 0.018$	0.018
$a\overline{b}c$	$\lambda_a\lambda_{\overline{b}}\lambda_c\theta_a\theta_{\overline{b}\|a}\theta_{c\|a}$	$0.6 \cdot 0.9 \cdot 0.4 = 0.216$	0
$a\overline{b}\overline{c}$	$\lambda_a\lambda_{\overline{b}}\lambda_{\overline{c}}\theta_a\theta_{\overline{b}\|a}\theta_{\overline{c}\|a}$	$0.6 \cdot 0.9 \cdot 0.6 = 0.324$	0
$\overline{a}bc$	$\lambda_{\overline{a}}\lambda_b\lambda_c\theta_{\overline{a}}\theta_{b\|\overline{a}}\theta_{c\|\overline{a}}$	$0.4 \cdot 0.3 \cdot 0.2 = 0.024$	0
$\overline{a}b\overline{c}$	$\lambda_{\overline{a}}\lambda_b\lambda_{\overline{c}}\theta_{\overline{a}}\theta_{b\|\overline{a}}\theta_{\overline{c}\|\overline{a}}$	$0.4 \cdot 0.3 \cdot 0.8 = 0.096$	0
$\overline{a}\overline{b}c$	$\lambda_{\overline{a}}\lambda_{\overline{b}}\lambda_c\theta_{\overline{a}}\theta_{\overline{b}\|\overline{a}}\theta_{c\|\overline{a}}$	$0.4 \cdot 0.7 \cdot 0.2 = 0.056$	0
$\overline{a}\overline{b}\overline{c}$	$\lambda_{\overline{a}}\lambda_{\overline{b}}\lambda_{\overline{c}}\theta_{\overline{a}}\theta_{\overline{b}\|\overline{a}}\theta_{\overline{c}\|\overline{a}}$	$0.4 \cdot 0.7 \cdot 0.8 = 0.224$	0

Now, $P(e) = WMC(\Delta) = 0.024 + 0.036 = 0.06$. The second manner of adding evidence to the knowledge base is by removing all those models that contradict the theory. This is achieved by computing $P(e) = WMC(\Delta \wedge \Delta_e)$, where Δ_e is the conjunction of the indicator variables that represent the evidence. In the case of the example with evidence $e = \{a, b\}$, $\Delta_e = \lambda_a \wedge \lambda_b$. The advantages of each of these two methods are addressed in Chavira and Darwiche [6].

Knowledge Compilation

One of the reasons that weighted model counting is an interesting approach to performing Bayesian network inference, is the fact that it facilitates a phenomenon known as *knowledge compilation*. While performing inference on real world sized Bayesian networks is generally intractable, representing the network in some other way may result in polynomial time algorithms to solve certain problems. Knowledge compilation is a fairly new research direction that deals with the intractability of general propositional reasoning. The basic idea is that some propositional theory, for instance a Bayesian network that is represented as a logical formula, as is explained in the previous section, is compiled into a target language in an *off-line phase*, after which a large number of queries can be answered in polynomial time in the *on-line* usage of the application. A target language is also a logical formula, but by means of putting some restrictions on the way sentences in the formula may be formed, some operations on the language can be performed much easier and faster. The restriction that can be placed onto logical formulas in order to arrive to a target language will be discussed in more detail in the next section. Compiling the original theory into some target language in an off-line phase offers the great advantage that a major part of the computational power needed for the theory is now dealt with in an off-line, rather than the on-line phase. This is particularly useful in mobile applications, since most mobile devices do not possess as much computational power as a desktop or laptop computer, meaning that any shift of computational steps from the on-line to an off-line phase should result in an increase in speed in everyday use of the application. Another advantage of compiling a given language into a different target language is that, given that the proper target language is chosen, certain queries are guaranteed to be answerable in polynomial time after compilation. This will be discussed in more detail in the following sections.

Compilation languages

As stated earlier, a number of restrictions can be imposed on a logical language in order to create a subset of that language that can be used to compile the original theory into. By imposing these restrictions, a representation of the original language is created that facilitates the possibility to perform certain operations very efficiently. These restrictions are extensively described in Darwiche and Marquis [5]. Any combination of the following restrictions can be imposed on the NNF language in order to obtain target languages for knowledge compilation:

1. **Flatness:** The height of NNF is at most 2.
2. **Simple Disjunction:** Every disjunction is a clause, where literals share no variables.
3. **Simple Conjunction:** Every conjunction is a term, where literals share no variables.
4. **Decomposability:** Conjuncts do not share variables.
5. **Determinism:** Disjuncts are logically disjoint.
6. **Smoothness:** Disjuncts mention the same set of variables.
7. **Decision:** A node of the form true, false, or $(X \wedge \alpha \vee \neg X \wedge \beta)$, where X is a variable and α, β are decision nodes.
8. **Ordering:** Decision variables appear in the same order on any path in the NNF.

A number of the target languages that are the result of imposing any number of these restriction on the NNF language can be found in the table in Figure 3.

Acronym	**Description**
NNF	Negation Normal Form
DNNF	Decomposable Negation Normal Form
d-DNNF	Deterministic Decomposable Negation Normal Form
sd-DNNF	Smooth Deterministic Decomposable Negation Normal Form
FBDD	Free Binary Decision Diagram
OBDD	Ordered Binary Decision Diagram
$\text{OBDD}_<$	Ordered Binary Decision Diagram (using order $<$)
DNF	Disjunctive Normal Form
CNF	Conjunctive Normal Form

Figure 3: A selection of the languages compared by Darwiche and Marquis.

The names of the target languages generally indicate the restrictions that are imposed on the NNF in order to arrive to that language. For each of the languages in the table in Figure 3 a short description is given, both to clarify the meaning of the restrictions and to give an impression of the resulting target language.

DNNF The target language DNNF is obtained by imposing decomposability on an NNF. This means that conjuncts do not share any variables. For instance,

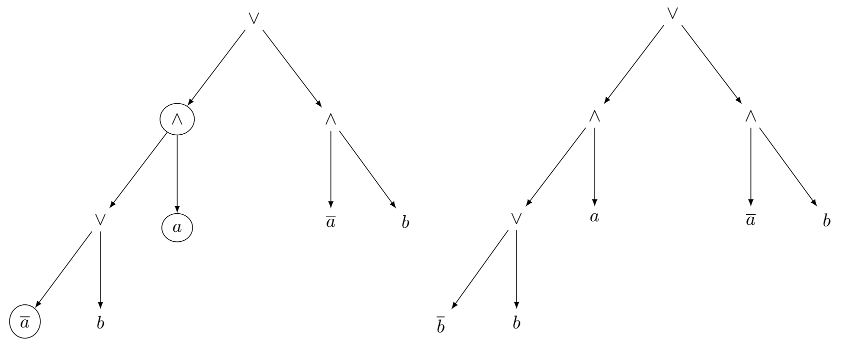


Figure 4: Example sentence.

the sentence depicted on the left in Figure 4 is not decomposable, since the conjuncts associated with the circled and-node both contain an instantiation of the variable a. The sentence on the right is decomposable, because none of the and-nodes have more than one child node with the same variable.

d-DNNF This language is obtained by imposing determinism on a DNNF. This means that disjuncts are logically disjoint. The sentence in Figure 5 is not deterministic, because the circled node has children $\neg a$ and b and these two are not disjoint, i.e. $\neg a \wedge b \not\models \bot$. For a sentence to be deterministic the children of an or-node may never be all true.

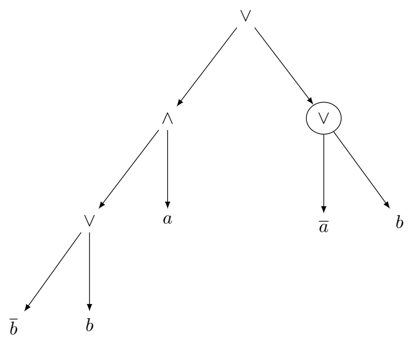
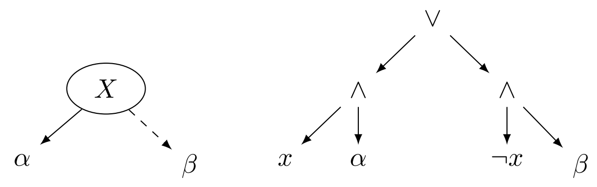


Figure 5: Example sentence.

sd-DNNF The target language sd-DNNF is a d-DNNF with the additional restriction that smoothness must be adhered to. A sentence is smooth if each disjunction in it mentions the same variables. The sentence in Figure 5 is not smooth either, as the circled node has a child with the variable A and a child with the variable B.

FBDD The way an FBDD is formed does not follow directly from its name. In order to understand what an FBDD is, a definition for *Binary Decision Diagrams (BDD)* is needed. BDD is the set of all NNF sentences, where the root of each (part of a) sentence is a decision node. Figure 6(a) shows a decision node as it is generally represented graphically. This node corresponds with the tree depicted in Figure 6(b) [5]. FBDD is the intersection of DNNF and BDD, in other words, it is a decomposable BDD.



(a) Decision diagram (b) Tree

Figure 6: A decision node (a) and its corresponding tree structure (b).

OBDD OBDD is FBDD with an additional ordering restrictions, meaning that all variables appear in the same order on all paths from the root to the leafs in the NNF.

The most important aspect when using knowledge compilation is choosing which target language the original theory will be compiled into. In order to do so Darwiche and Marquis [5] propose three different key properties of compilation languages, which can be utilized in order to make an informed decision on the target language most suited for a particular type of application. These key properties are:

1. **Level of succinctness:** Let L_1 and L_2 be two subsets of NNF. L_1 is at least as succinct as L_2 , denoted $L_1 \leq L_2$, iff there exists a polynomial p such that for every sentence $\alpha \in L_2$, there exists an equivalent sentence $\beta \in L_1$ where $|\beta| \leq p(|\alpha|)$.

2. **Set of queries supported in polynomial time:** Checking for consistency, validity, clausal and sentential entailment, implicant and equivalence, as well as model counting and model enumeration.

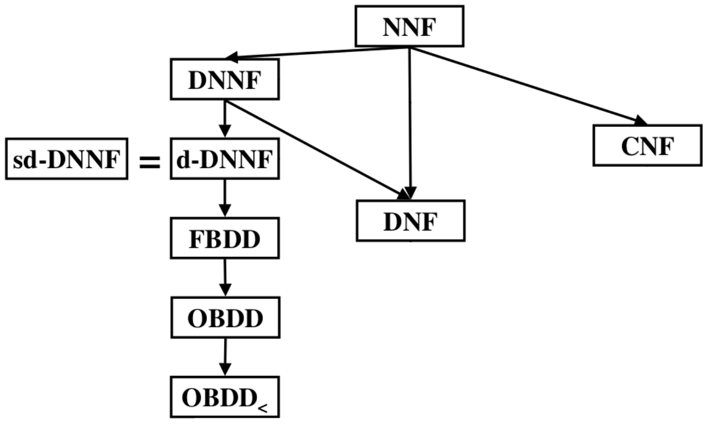


Figure 7: A graph representing the succinctness of a number of subsets of the NNF language as presented by Darwiche and Marquis [6]. An edge $L_1 \rightarrow L_2$ indicates that L_1 is strictly more succinct than L_2: $L_1 < L_2$, while $L_1 = L_2$ indicates that L_1 and L_2 are equally succinct: $L_1 \leq L_2$ and $L2 \leq L_1$. Dotted arrows indicate unknown relationships.

3. **Set of transformations supported in polynomial time:** Conditioning, (singleton) forgetting, (bounded) conjunction, (bounded) disjunction and negation.

The tables in Figure 8 and 9 give an overview of the queries and transformations discussed in [5].

Of course, the exact meaning of each of the table entries in both these tables is not included. As such, each query and transformation will be introduced briefly.

A language L supports *polytime consistency checking (CO)* if there exists a polynomial time algorithm to determine whether any formula in L is consistent or not. Consistency in this context refers to logical consistency, i.e. the formula does not contain any contradictions. *polytime validity checking (VA)* is very similar to CO in the sense that L support VA if there exists a polytime algorithm to determine if any formula in L is valid, where validity means that the formula is true under every interpretation. L supports *polytime causal entailment checking (CE)* if there exists an algorithm that checks whether γ is entailed by Σ ($\Sigma \models \gamma$) for all clauses γ and all formulas Σ from L in polynomial time. If it is possible to check whether $\gamma \models \Sigma$ in polynomial time, L satisfies *polytime implicant checking (IM)*. If L satisfies *polytime equivalence checking (EQ)* if there exists a polynomial time algorithm that determines whether $\Sigma \equiv \Phi$ holds for any pair of formulas Σ and Φ in L. If there exists such an algorithm to determine whether

Notation	Query
CO	polytime consistency check
VA	polytime validity check
CE	polytime clausal entailment check
IM	polytime implicant check
EQ	polytime equivalence check
SE	polytime sentential entailment check
CT	polytime model counting
ME	polytime model enumeration

Figure 8: An overview of notations for queries.

Notation	Transformation
CD	polytime conditioning
FO	polytime forgetting
SFO	polytime singleton forgetting
$\wedge$**C**	polytime conjunction
$\wedge$**BC**	polytime bounded conjunction
$\vee$**C**	polytime disjunction
$\vee$**BC**	polytime bounded disjunction
$\neg$**C**	polytime negation

Figure 9: An overview of notations for transformations.

$\Sigma \models \Phi$, L satisfies *polytime sentential entailment checking(SE)*. L satisfies *polytime model counting (CT)* if there is a polytime algorithm to determine the number of models for each sentence in L. Finally *polytime model enumeration (ME)* is satisfied by L if there exists a polynomial $p(n, m)$, where n is the size of some formula in L and m is the number of models for that sentence, such that all models for the sentence can be output in time $p(n, m)$.

A language L satisfies *polytime bounded conjunction ($\wedge$BC)* if every pair of formulas in L can be mapped to a formula in L that is equivalent to the conjunction of these two formulas. If the same can be done for any finite set of formulas in L, L satisfies *polytime conjunction ($\wedge$C)*. Conversely, L satisfies *polytime bounded disjunction ($\vee$BC)* if pair of formulas can be mapped to a formula equivalent to the disjunction of these formulas. Again, if the same can be done for any finite set of formulas in L, the language satisfies *polytime disjunction ($\vee$C)*. If every formula in L can be transformed into another formula that is equivalent to the negation of that formula, L is said to satisfy *polytime negation ($\neg$C)*. If, for every formula Σ in L and every term γ, each variable X of Σ can be replaced by true if x is a literal of γ and by false if $\neg x$ is a literal of γ in polytime, L satisfied *polytime conditioning (CD)*. Finally, L satisfies *polytime forgetting (FO)* if for every set of variables $\mathbf{X}$ and every sentence Σ in L a sentence Σ' in L can be constructed such that for every formula α that does not mention any variable in $\mathbf{X}$ $\Sigma \models \alpha$ holds precisely when $\Sigma' \models \alpha$ holds. If this property holds, but merely for a single variable rather than a set of variables, L satisfies *polytime singleton forgetting (SFO)*.

The main goal when selecting the appropriate target language for a project is to first determine which queries and transformations should be supported in polynomial time and when this is done, choosing the most succinct language to support these features. In the next section an overview is presented of all the queries and transformations that are supported in polytime by each of the languages mentioned, as well as their succinctness.

The SDD target language

Besides the target languages discussed in the previous section, another, newer target language is examined as a candidate for usage in mHealth applications that use Bayesian network inference: the *Sentential Decision Diagram* or *SDD*. This language was proposed in 2011 by Darwiche [7] and as such was not a part of the original comparison between target languages. Because of this, and the fact that this language possesses characteristics that allow for fast compilation into a compact representation (given a good heuristic) [7], an in depth analysis of this target language is made in order to include SDD comparison between target languages.

SDD is the language that is obtained by imposing two newer restrictions on the NNF language: *structured decomposability* [17] and *strong determinism* [18]. If a language adheres to structured decomposability, it adheres to the decomposability restriction discussed earlier but is also structured. A structured language is a language that respects a *vtree*, where a vtree for a set of variables $\mathbf{X}$ is defined as a full, rooted binary tree whose leaves are in one-to-one correspondence with the variables in $\mathbf{X}$. A language is strongly deterministic if the conjunction of the formula represented by any pair of nodes in the vtree is inconsistent. As said, SDD adheres to both these restrictions and is a strict subset of d-DNNF and a strict superset of OBDD [7]. The question here is how exactly it compares to these two, but also all the other target languages discussed. To answer this question the queries and transformations supported by SDD in polytime are discussed, as well as its succinctness in comparison to the other languages discussed in this thesis.

Queries and Transformations

The first property on basis of which SDD will be compared to the other languages are the queries and transformations that are supported in polynomial time this target language. These properties have recently been discussed extensively by Van den Broeck and Darwiche [9]. These results have been added to the comparison table presented in [5] and can be seen in Figures 10 and 11.

	CO	VA	CE	IM	EQ	SE	CT	ME
NNF	○	○	○	○	○	○	○	○
DNNF	✓	○	✓	○	○	○	○	✓
d-DNNF	✓	✓	✓	✓	?	○	✓	✓
sd-DNNF	✓	✓	✓	✓	?	○	✓	✓
SDD	✓	✓	✓	✓	✓	○	✓	✓
FBDD	✓	✓	✓	✓	?	○	✓	✓
OBDD	✓	✓	✓	✓	✓	○	✓	✓
$\text{OBDD}_<$	✓	✓	✓	✓	✓	✓	✓	✓
DNF	✓	○	✓	○	○	○	○	✓
CNF	○	✓	○	✓	○	○	○	○

Figure 10: A table representing the queries supported in polytime by each of the languages as presented by Darwiche and Marquis, with the addition of the SDD language. ✓means that the query is supported by the language in polytime, whereas ○means it is not, unless P=NP.

	CD	FO	SFO	$\wedge$C	$\wedge$BC	$\vee$C	$\vee$BC	$\neg$C
NNF	✓	○	✓	✓	✓	✓	✓	✓
DNNF	✓	✓	✓	○	○	✓	✓	○
d-DNNF	✓	○	○	○	○	○	○	?
sd-DNNF	✓	○	○	○	○	○	○	?
SDD	✓	●	✓	●	○	●	○	✓
FBDD	✓	●	○	●	✓	●	✓	✓
OBDD	✓	●	✓	●	○	●	○	✓
$\text{OBDD}_<$	✓	●	✓	●	✓	●	✓	✓
DNF	✓	✓	✓	●	✓	✓	✓	●
CNF	✓	○	✓	✓	✓	●	✓	●

Figure 11: A table representing the transformations supported in polytime by each of the languages as presented by Darwiche and Marquis, with the addition of the SDD language. ✓means that the query is supported by the language in polytime, ●means it is not supported and ○means it is not supported unless P=NP.

Succinctness

Besides the operations that are supported in polynomial time, the succinctness of the SDD target language needs to be analyzed. Because this is not discussed in previous literature, an analysis of the succinctness property with regards to SDD is presented here.

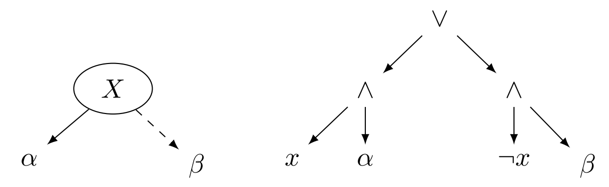
Because SDD is a proper subset of d-DNNF, d-DNNF $\leq$ SDD holds. Conversely, because SDD is a strict superset of OBDD, SDD $\leq$ OBDD holds as well. Because the succinctness relation adheres to transitivity we can now conclude that for all target languages L for which $L \leq$ d-DNNF holds, $L \leq$ SDD holds as well. Conversely for all languages L' which adhere to OBDD $\leq L'$, SDD $\leq L'$ also holds. As can be seen in Figure 7, FBDD is the only language positioned in between d-DNNF and OBDD concerning succinctness. To determine where SDD stands compared to FBDD in terms of succinctness, it makes sense to look at the properties of both languages. In the following, we make use of the following two results. The first..

Theorem 1 ([8])**.** *All Boolean functions that can be represented by a tree structured circuit, can be represented by an SDD whose size is linear the size of the circuit.*

Conversely Breitbart et al. have proved that there exist Boolean functions that can only be represented by an FBDD that is exponential in the number of variables:

Theorem 2 ([19], Theorem 6)**.** *For every $n \geq 4$, there exists a Boolean function Φ, such that every FBDD computing Φ contains at least 2^n nodes, but there is a BDD computing Φ with no more than $\mathcal{O}(n^2)$ nodes.*

This means that if every FBDD can be represented by a tree structured circuit, we can say something about the succinctness of the FBDD language compared to the SDD language.



(a) Decision diagram (b) Tree

Figure 12: A decision node (a) and its corresponding tree structure (b).

Definition 1. *A decision node in an BDD as depicted in Figure 12(a) corresponds to the tree structure depicted in Figure 12(b) [5].*

Proposition 1. *For all $n \geq 0$ it holds that a BDD with n decision nodes can be represented by a tree with $6n + 1$ nodes.*

Proof. Proof by complete induction on n.
If $n = 0$, then the BDD consists of a single node representing 0 or 1. The associated tree is exactly the same, i.e., the tree has 1 node.
Let m be any natural number ≥ 0 and assume that the proposition holds for all BDDs with $0 \leq i \leq m$ decision nodes (induction hypothesis). A BDD with $m + 1$ decision nodes can be represented as the tree depicted in Figure 12(b), where both α and β represent subtrees with k and l decision variables such that $k+l = m$. This tree then has $5+(6k+1)+(6l+1)(\text{IH}) = 6m+7 = 6(m+1)+1$ nodes. □

Proposition 2. *SDDs are at least as succinct as FBDDs, i.e.,* $SDD \leq FBDD$.

Proof. Because all BDDs of size n can be represented by a tree of size $6n + 1$ (Proposition 1), the same holds for all FBDDs, since $FBDD \subseteq BDD$. By Theorem 1, it follows that all these FBDDs can be represented by an SDD linear in n. □

Proposition 3. *SDDs are strictly more succinct than FBDDs, i.e.,* $SDD < FBDD$.

Proof. From Proposition 1 and Theorem 2 it follows that there is a function that can only be computed by an FBDD with a tree of size $\mathcal{O}(2^n)$ that be represented by a different BDD with a tree of size $\mathcal{O}(n^2)$ and thus by an SDD of size $\mathcal{O}(n^2)$ (Theorem 1). Therefore, it holds that $FBDD \not\leq SDD$. Together with Proposition 2, the property follows. □

	NNF	DNNF	d-DNNF	sd-DNNF	SDD	FBDD	OBDD	OBDD$_<$	DNF	CNF
NNF	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$
DNNF	$\not\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\not\leq$
d-DNNF	$\not\leq$	$\not\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\not\leq$	$\not\leq$
sd-DNNF	$\not\leq$	$\not\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\not\leq$	$\not\leq$
SDD	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\leq$	$\leq$	$\leq$	$\leq$	$\not\leq$	$\not\leq$
FBDD	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\leq$	$\leq$	$\leq$	$\not\leq$	$\not\leq$
OBDD	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\leq$	$\leq$	$\not\leq$	$\not\leq$
OBDD$_<$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\leq$	$\not\leq$	$\not\leq$
DNF	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\leq$	$\not\leq$
CNF	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\not\leq$	$\leq$

Figure 13: succinctness table with SDD

4 Analysis

Now that all properties for all target languages have been established, the next step is to determine what queries and transformations are important in an mHealth application that uses Bayesian network inference. To do so, all queries and transformations will be discussed and a decision will be made on whether or not it needs to be supported by the ideal target language.

Queries

In mHealth applications that use Bayesian network inference, the improvements from compiling the network are mainly to be made in the time needed to perform inference based on some evidence.

(CO, VA) In general, any mHealth application that utilizes Bayesian network inference will use a set network to perform inference on. Because a Bayesian network is always consistent, it is of no importance whether or not a check for consistency is supported in polynomial time. A polytime validity check is not important for mHealth applications that use Bayesian network inference, as the need for all models to be true will never arise. If all models in a Bayesian network are true, the probability for a query would simply be 1. If other types of probabilistic models, such as more general probabilistic logical models are taken into account, consistency or validity checks may become more important. However, it makes sense that in such cases the model is created in an off-line phase and that consistency and validity can be checked in this phase as well. Once the model has been created, there will generally be no need to adjust the network on the mobile device itself. It is therefore not important whether the target language supports polytime consistency or validity checking, even if other probabilistic models are taken into account.

(CE, IM, EQ, SE) These four queries are grouped together because all deal with entailment in some form. The main usage of all of these queries is to check for equivalence of sentences[5]. Equivalence checks are important when multiple representations of the same theory are proposed and it needs to be determined if both are equal. However, in mHealth applications this situation will never occur in the on-line phase of the application. It might be the case that two representations for some theory are proposed. However, before any application is deployed to the mobile device, the proper representation should already be decided on. As such it is not important for mHealth applications whether clausal entailment, implicant, equivalence and sentential entailment can be checked in polynomial time.
Another usage of these queries might again be in the case that some other logical model is used, rather than one created from a Bayesian network. In general logical models it may be desirable to be able to check whether or not some clause or formula is entailed by another.

(CT) As explained earlier in this thesis, model counting is the method to determine the probability for a given statement when used for Bayesian network inference. In other words, the probability for the value of some variable X given evidence e can be calculated by means of model counting. This probability is equal to the weighted sum of all models in which the statement is true. Determining these types of probabilities is the main goal of any mHealth application that uses Bayesian networks. As such this query needs to be supported in polytime by any target language which is to be used in an mHealth application.

(ME) The polytime model enumeration property is adhered to if there exists a polynomial $p(n, m)$ over the size n of some input sentence and the number of models m for that sentence. The problem with this is that a polynomial time algorithm is not guaranteed with this, as the size of the input sentence could be exponential, making the p an exponential function as well. As such this query is not particularly useful in an mHealth application that uses Bayesian network inference. However, all target languages that support polytime model counting must also support model enumeration. As polytime model counting is a necessary condition in any application in any mHealth application that utilizes Bayesian network inference, polytime model enumeration will be supported by default by any suitable target language.

Transformations

(CD) Besides model counting, this is the most important property that a target language used for an mHealth application based on Bayesian network inference should support in polynomial time. Conditioning is used to set the evidence in the compiled theory, by setting each variable in the theory that is part of the evidence to its corresponding truth value. Because the main purpose of any mHealth application that uses a Bayesian network would be to determine some risk or probability, given some evidence, this transformation must be supported in polytime.

(FO, SFO) Both forgetting as well as singleton forgetting might be useful in situations where the entire network is not always needed. For instance in a situation where a number of variables that are part of the Bayesian network used are only applicable to women. In that case it would make sense that if the user of the application is male, those variables that are not applicable to males can be forgotten from the network to improve memory usage and time needed for inference. The question then is whether or not this transformation should be done in the on-line or the off-line phase. In the case that only a single application, rather than two distinct applications for men and women, is released, the on-line phase will generally be favored. This will most likely only be applicable to a small fragment of all mHealth applications and even when it is, it is the question whether forgetting part of the network truly offers a great advantage in practice. Therefore polytime (singleton) forgetting should be considered when choosing the appropriate target language, yet it should be given a relatively low weight.

($\wedge$C, $\wedge$BC, $\vee$C, $\vee$BC, $\neg$C) These transformations all deal with transforming some set ($\wedge$C, $\vee$C, $\neg$C) or pair ($\wedge$BC, $\vee$BC) of sentences in the target languages into a single sentence that is respectively the conjunction, the disjunction or the negation of these sentences. All these properties have some value when building a compiler into a target language, meaning that polytime support for these transformations might be important when a compiler needs to be implemented.

However, this is not the aim of this thesis. Rather the aim is to determine what properties are important in the on-line phase in an mHealth application. Therefore, even though these transformations are important in many situations, these operations are of minor importance in the final application.

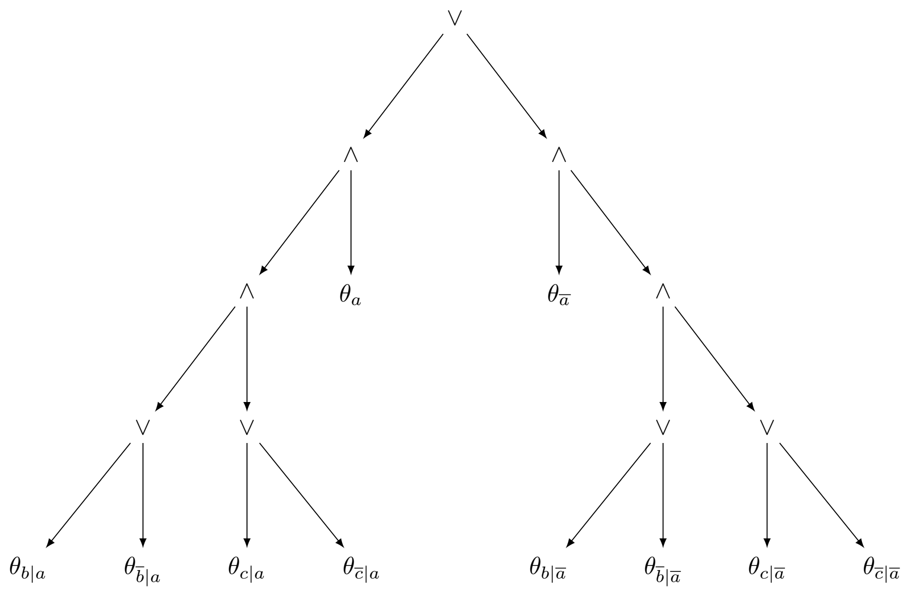
Most suited target language

Given that we have now established the queries and transformations that should be supported by the ideal target compilation language, we can now decide on the theoretical best choice for a compilation language. The most important of these operations are model counting and conditioning. As stated, some operations might be valuable for certain distinct situations. However, as the aim of this thesis is to find the most suited target language for mHealth applications in general, the most succinct language that would support the majority of mHealth applications is chosen. As such the most succinct language that supports both polytime model counting as well as polytime conditioning is considered the theoretical best choice. The table in Figure 13 indicates that d-DNNF or sd-DNNF would be the theoretical best choice as these two languages are the most succinct languages to support the essential operation in polynomial time. SDD should also be mentioned as a viable candidate for a target language for mHealth applications that use Bayesian network inference. Though not as succinct as d-DNNF and sd-DNNF, this language has the additional advantage that it supports a lot more transformations in polynomial time. Because of this, compilation into this language is more efficient than it is for d-DNNF and sd-DNNF [7]. Especially for extremely large networks this is something that should be taken into consideration. However, because d-DNNF and sd-DNNF are the more succinct languages, these two languages are considered to be the most suited target languages for the purpose of Bayesian network inference. Though these two languages are ranked equally, in the following sections d-DNNF is chosen as the most suited target language, for the simple fact that it is the more practical choice of these two languages, as a stable compiler for it has already been implemented.

Bayesian network to d-DNNF

Because d-DNNF is the most suitable target language for mHealth applications that use Bayesian network inference, a small example on how probabilities can be obtained from a compiled network is given. Figure 14 shows the what the example network used earlier would look like of it were compiled into d-DNNF form. The representation in the graph shown is a simplified version of the true d-DNNF, as the indicator variables (λ) have been left out. As such the children of or-nodes in the tree are not logically disjunct. We can however assume that those variables whose subscripts are disjoint are logically disjoint.

The easiest way to determine probabilities in this network is by transforming the d-DNNF into an arithmetic circuit, which is achieved by replacing all and-nodes by a multiplication and all or-nodes by an addition and adding all



$\vee$
$\wedge$
$\wedge$
$\wedge$
θ_a
$\theta_{\overline{a}}$
$\wedge$
$\vee$
$\vee$
$\vee$
$\vee$
$\theta_{b|a}$
$\theta_{\overline{b}|a}$
$\theta_{c|a}$
$\theta_{\overline{c}|a}$
$\theta_{b|\overline{a}}$
$\theta_{\overline{b}|\overline{a}}$
$\theta_{c|\overline{a}}$
$\theta_{\overline{c}|\overline{a}}$

Figure 14: The example network as a d-DNNF.

probabilities to the variables.

This would result in the tree that can be seen in Figure 15. Say that B is observed as true. The network is now conditioned on the evidence b, which means that all variables whose subscript contradicts b are assigned a weight 0. The result of this conditioning is displayed in Figure 16. After conditioning on b, the entire network now evaluates to 0.18, which is $P(b)$. Say we now want to know the probability that A true given the evidence. This can be achieved by conditioning the network in Figure 16 on a, which would result in the entire right subtree evaluating to 0. The entire network now evaluates to 0.6. In order to now determine the probability $P(a \mid b)$ we simply divide these to evaluations to get to 0.33.

5 The eMomCare Project

In order to answer the second research question "What are the improvements that can be achieved by utilizing this [best suited] target language in an existing application?", a test application will be implemented based on an existing mHealth application developed at the Radboud university named eMomCare.

Pre-eclampsia

The eMomCare Project is an mHealth application aimed specifically towards pregnant women[3]. A common complication in pregnancy is a syndrome called

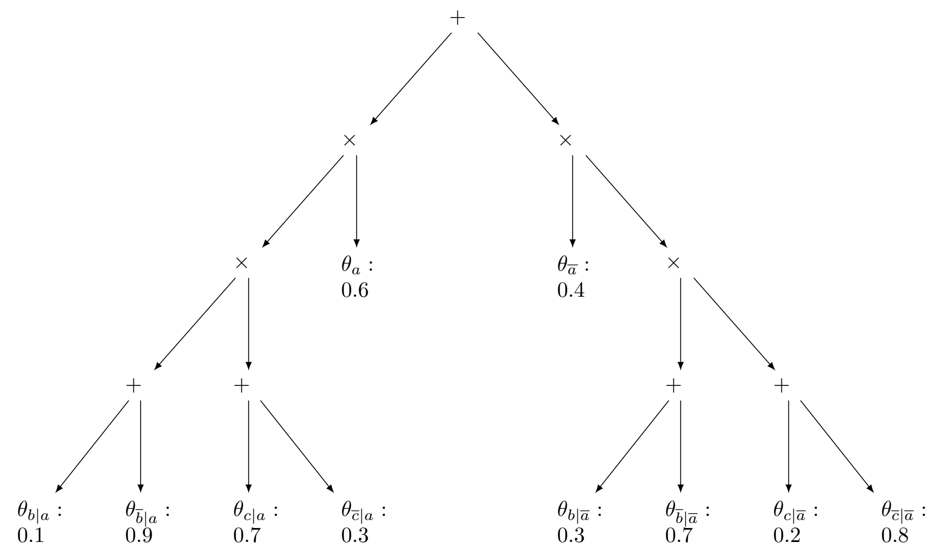

+
×
×
×
θ_a : 0.6
$\theta_{\overline{a}}$: 0.4
×
+
+
+
+
$\theta_{b|a}$: 0.1
$\theta_{\overline{b}|a}$: 0.9
$\theta_{c|a}$: 0.7
$\theta_{\overline{c}|a}$: 0.3
$\theta_{b|\overline{a}}$: 0.3
$\theta_{\overline{b}|\overline{a}}$: 0.7
$\theta_{c|\overline{a}}$: 0.2
$\theta_{\overline{c}|\overline{a}}$: 0.8

Figure 15: The d-DNNF as an arithmetic circuit.

pre-eclampsia, which often occurs after 20 weeks of pregnancy or immediately following the delivery. The syndrome can affect the pregnant woman's kidneys, liver, heart, and brain and is diagnosed in approximately 7.5% of first time pregnant women. In the Netherlands, it is the most important cause of death among pregnant women and because forcing the (early) delivery of the baby is the only cure for pre-eclampsia, it is important to identify those with a high risk of suffering from this syndrome as early as possible. If high risk is detected in an early stage, anti-hypertensive treatment can be used to reduce the risk on pre-eclampsia.

The project

The eMomCare system is a mobile home-monitoring system which can be used by pregnant women to help determine their risk of developing pre-eclampsia. Many of the data needed to diagnose pre-eclampsia, but also to predict the risk thereof, can be obtained by measurements done by the patients themselves. These data are then automatically sent to the health-care team. This offers multiple advantages over the traditional method of periodical checkups performed at the medical doctor's office. The first advantage is that the measurements taken in a domestic setting are often a more accurate representation of the everyday values than those taken in a clinical setting. This especially holds true for the measurement of blood pressure, which is known to generally be elevated compared to its normal value in a clinical environment, a phenomenon known as the white-coat effect. Secondly, the patient will generally need to visit the hospital less frequently and also be more actively involved in the monitoring of

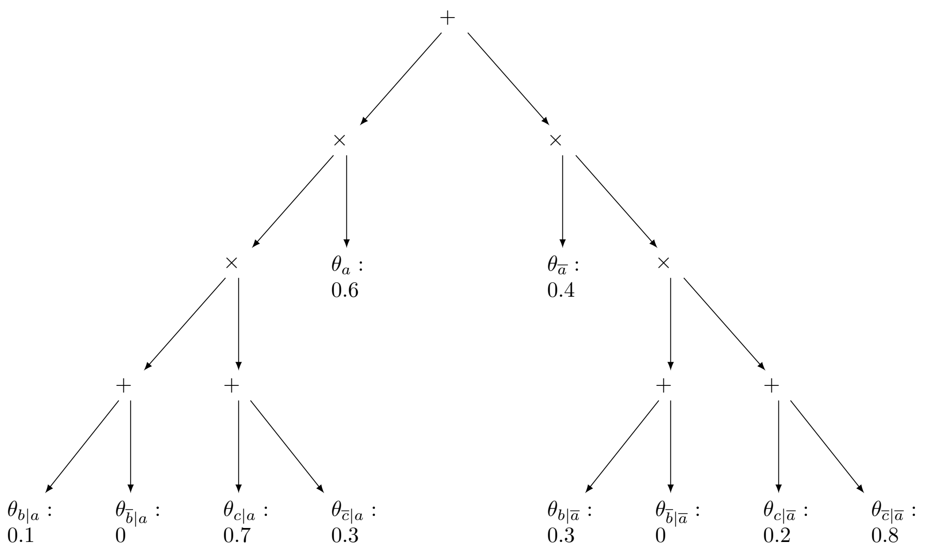

+
×
×
×
θ_a : 0.6
$\theta_{\overline{a}}$: 0.4
×
+
+
+
+
$\theta_{b|a}$: 0.1
$\theta_{\overline{b}|a}$: 0
$\theta_{c|a}$: 0.7
$\theta_{\overline{c}|a}$: 0.3
$\theta_{b|\overline{a}}$: 0.3
$\theta_{\overline{b}|\overline{a}}$: 0
$\theta_{c|\overline{a}}$: 0.2
$\theta_{\overline{c}|\overline{a}}$: 0.8

Figure 16: The arithmetic circuit conditioned on b.

her pregnancy. This in turn leads to another possible advantage: the work load for the obstetric care, as well as the cost for healthcare, can be reduced.

Technical realization

In order to monitor the patient sufficiently and more importantly, to make informed decisions as well as getting an accurate prediction for the risk of preeclampsia occurring, a number of different technical aspects of the system come into play. The risk is determined using the following:

1. Collection of patient and sensor data. This is done by means of questionnaires, automatic reading of measurement equipment such as electronic blood pressure meter via Bluetooth and automatic analysis of urine strips using the phones camera and image processing techniques.

2. Automatic interpretation of both patient and sensor data within the smartphone itself by a specially designed Bayesian network. The model can be used to provide feedback, explain the results obtained and recommend actions to the patient and the care team regarding the progression, or lack thereof, of the syndrome.

3. Communication of the results, both textually and visually, to the care team and the patient. The data should be stored in a hospital database for further inspection by the caregivers.

Network used

The eMomCare system is based on a Bayesian network designed specifically for the project. The network was designed to take all clinical knowledge concerning pre-eclampsia into consideration.

Queries in eMomCare

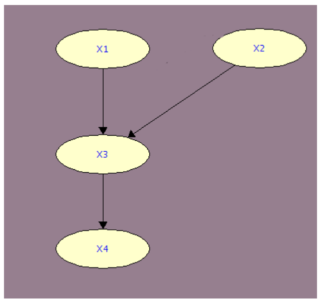
To demonstrate that the queries supported by the selected target language d-DNNF are sufficient for usage in a practical application, the queries required to improve eMomCare are discussed briefly. The aim in this application is to improve the speed with which inference can be performed. In order to achieve this increase in speed, the compilation language should at least support polytime model counting. All other queries mentioned are of lesser importance for the eMomCare project, simply because most of these queries are useful if changes have been made to the model. However in the case of this project, this is not a likely scenario. The model used will not be changed often, presumably merely if new clinical data is found that necessitates a change or addition to the existing Bayesian network. If this is the case however, it would mean that the Bayesian network itself is modified, not the compiled theory. The modified network would then be compiled to a new compiled theory, making the need for the target language to support any of the queries other than model counting in polytime small to non-existent.

Transformations in eMomCare

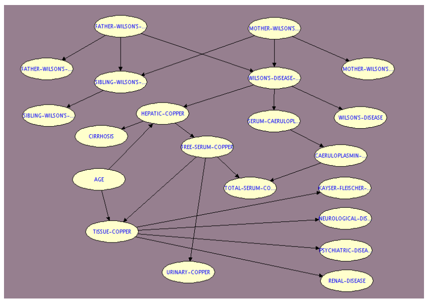
For the same reason that nearly all queries need not be supported by the target language, none of the transformations are of importance for the eMomCare project. All transformations are useful for changing something in the model in the on-line phase. As explained earlier, this does not offer any additional value in this project.

6 Testing

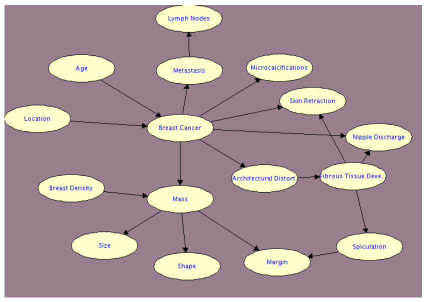
In order to test the performance of the compiled network compared to the original network with regards to the time needed to perform inference, two different kinds of tests were run. The first test consisted of performing inference on five different Bayesian networks, all of different sizes, including the network used in the eMomCare project. Figures 17 and 19 show (a part of) the networks used for this test. For all networks both a compiled and an uncompiled version of the network were used in the test and in all cases the evidence and query variables were the same in both versions of the network. In the two largest networks (E4 and the eMomCare network), tests with more than one query variables were run as well. This test was performed to measure the performance of the original networks and the compiled network based on input size. The results for the test are displayed in the table in Figure 21 and are discussed in the results section.



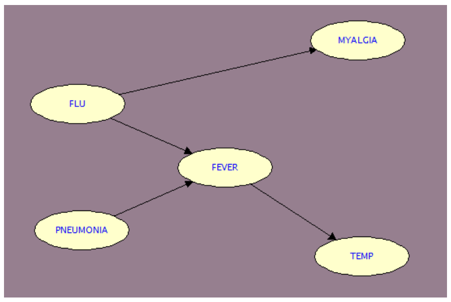
(a) E1



(b) E4



(c) E3



(d) E2

Figure 17: Testing networks used

The times in the table are averages over a hundred trials.

The second test consisted of a large number of queries on the eMomCare network. These were all queries that are indicative of the types of queries that the full application should support in everyday use. The following queries were examined:

- P(PE$_{WEEK}$=yes),$_{WEEK}$=12,16,20,24,28,32,36,38,40,42, based on a number of risk factors as found in Figure 18. These numbers provide the baseline for the risk of developing pre-eclampsia without any measured signs.
- P(PE$_{WEEK}$=yes),$_{WEEK}$=12,20,32,42, based on the previously established risk factors and a number of signs for $_{WEEK}$. The probabilities were determined for all different values for treatment$_{WEEK}$.
- P(PE$_{WEEK+}$=yes), with the same conditions as the previous bullet, and $_{WEEK+}$ representing the checkup following $_{WEEK}$.
- The expected values for the signs in $_{WEEK+}$.

Type	Factor	Abbreviation	Values / Ranges
Risk factors	Antiphospholipid syndrome	APS SYNDROME	no, yes
	Parity and History of preeclampsia	PARITY-HISTORYPE	nulliparous, parous-yes, parous-no
	Chronic hypertension	CHRONIC HT	no, yes
	Renal disease	RENALDISEASE	no, yes
	Diabetes	DIABETES	no, yes
	Drugs for Chronic hypertension	TREATMENT-CHT	no, yes
	Drugs for Renal disease	TREATMENT-RD	no, yes
	Drugs for Diabetes	TREATMENT-DB	no, yes
	Family history of preeclampsia	FH-PE	no, fatherPEpreg, mother-sister
	Family history of hypertension	FH-HT	no, yes
	Family history of diabetes	FH-DIAB	no, yes
	Multiple pregnancy	MULTI-PREGNANCY	no, yes
	Obesity	OBESITY	normal, overweight, obese
	Maternal age	AGE	< 20, 21-25, …, > 40
	Smoking	SMOKING	no, yes
Signs	Systolic blood pressure (mmHg)	SBP	< 109, 110-119,…, 160-169, > 170
	Diastolic blood pressure (mmHg)	DBP	< 59, 60-69,…, 100-109, > 110
	Hemoglobin (mmol/L)	HB	6.2, 6.3, …, 9.3
	Creatinine (μmol/L)	CREAT	< 45, …,118-121,> 122
	Protein (Albumin)–Creatinine ratio	PACR	0-0.03, 0.04-0.06, …, 4.5-5, > 5
Extern.	Drugs taken by the patient	TREATMENT	no, Anti-HT, Other, Anti-HT+Other
Hidden	Vascular risk	VASCRISK	false, true
	Vascular function	VASCFUNC	hypotens., normal, hypertens., severe-hypertens.
	Renal function	RENALFUNC	ok, nok
Syndr.	Preeclampsia	PE	no, yes

Figure 18: An overview of the variables in the eMomCare network

Firstly a number of Risk Factors, as they are included in the original application was set. With these risk factors, the baseline for the risk for the development of pre-eclampsia during the pregnancy was calculated using both the compiled, as well as the original network. This baseline was then plotted as a graph in the results for both versions of the network in order to quickly identify potential differences in the output probabilities. In theory the outputs for both versions should be identical. In a second step different signs were added for the and the effect of treatment for the weeks 12, 20, 32 and 42. These results were also plotted in the results. The third step was to predict the values for the week after the current week (i.e. the week for which the signs were added). Finally the expected values for the signs at the next checkup were calculated and displayed in the results.

Implementation

To produce the d-DNNF to be used for inference in the test networks, UCLA's Ace compiler [20] was used. For the implementation in the application the on-line engine code provided with the Ace package was used as a basis. The choice was made to use a simple test application with a minimal user interface. A separate parser class was used to generate most of the code for the layout of the test application. Because the eMomCare application currently uses the EBayes [21] library for inference, this library was used to provide the data with which the compiled network was compared. In order to display the probabilities calculated in the application, a graph view was used. The rendering of the graphs was implemented using the GraphView Library [22]. All of these individual components were combined in the test application and adjusted for usage on a mobile device where needed. The main drawback of the current implementation is that the entire compiled network is loaded into Java on the startup of the application. In theory, this should be done in the off-line phase, however for the purposes of the test application, rendering the network on-line is not a problem. In the final application this should be changed.

Results

Compilation time

Before either of the two previously described tests could be performed, the original Bayesian networks needed to be compiled into d-DNNF form. As said earlier, UCLA's Ace compiler was used to do so. Even though the most important aspect of a target language that is to be used in an mHealth application that uses Bayesian network inference is its performance in the on-line phase, it still is interesting to look at the time needed for compilation to determine whether or not the possible improvements in the on-line phase are large enough to justify the compilation. The time needed for compilation for each of the test networks can be found in the table in Figure 20. The table shows all individual components that are involved in the compilation process. While the compile

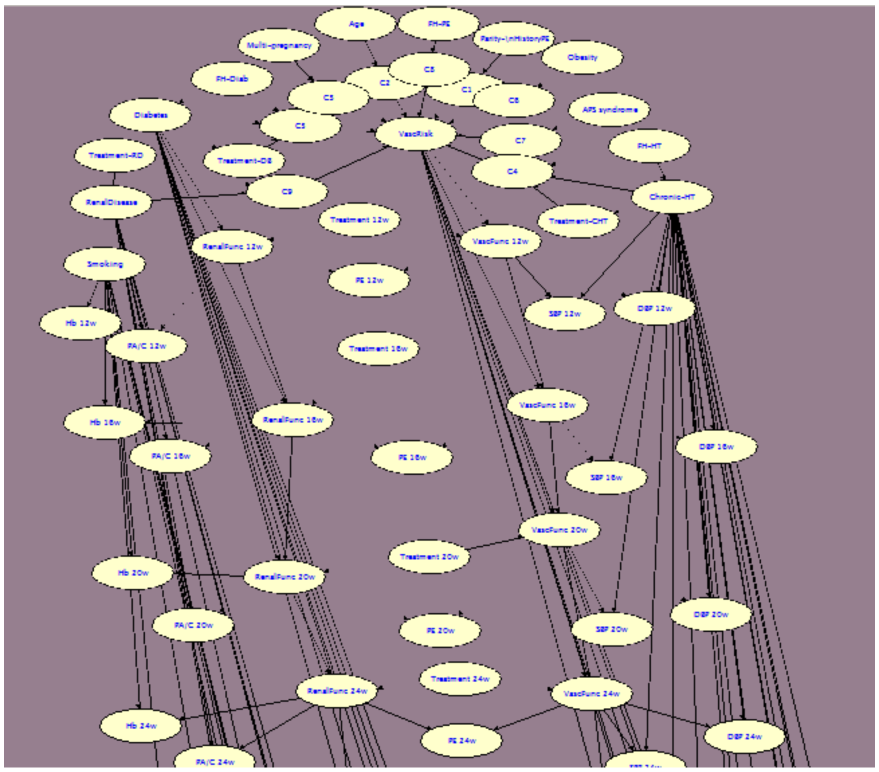


Figure 19: Part of the eMomCare network

time may be the most important aspect of these components, it accounts for a rather small portion of the total time needed for compilation, especially in the smaller networks. It is for this reason that it makes sense to look at the total time for the compilation process, rather than just the time needed for compilation only. It appears this total time increases with the size of the input network, but not in an exponential fashion.

The first test

The results for the first test can be found in Figure 21. For small networks the original and the compiled network perform comparatively well. However, as the input size increases, the inference time for the original network increases a lot faster than that of the network compiled to a d-DNNF. Because both versions of the network show a similar performance for small inputs, the large discrepancy between the two for larger networks can only be explained by the reduction in complexity for inference in the compiled network.

Network	**Nodes**	**Edges**	**Network read time**	**Initialization time**	**Compilation time**	**Write time**	**Total time**
E1	4	3	167ms	37ms	3ms	4ms	213ms
E2	5	4	168ms	40ms	3ms	4ms	221ms
E3	16	18	150ms	43ms	5ms	6ms	204ms
E4	21	23	163ms	51ms	5ms	7ms	227ms
eMom	105	186	296ms	124ms	41ms	81ms	543ms

Figure 20: Compilation times for the test networks.

Network	**Nodes**	**Edges**	**Query variables**	**Original**	**d-DNNF**
E1	4	3	1	<1ms	<1ms
E2	5	4	1	<1ms	<1ms
E3	16	18	1	∼4ms	<1ms
E4	21	23	1	∼11ms	<1ms
E4	21	23	4	∼13ms	<1ms
eMom	105	186	1	∼293ms	∼43ms
eMom	105	186	10	∼2599ms	∼45ms

Figure 21: Results for the first test. All results are averages over 100 trials

The second test

In the second test the advantage of the d-DNNF over the original network became even more apparent. The entire second test ran in approximately 600ms on the compiled network, averaged over ten trials. The second test could not be completed using the original network due to out of memory errors. To still test the performance of both networks, the queries for the final week (week 42) were left out. The result was that the compiled network again ran in circa 600ms on average, whereas the original network needed approximately 185 000ms. Though many of these increases in speed can be explained by the theory presented in this thesis, there is another, more practical aspect that influences the time needed to complete the second test. Because of the many variables involved in the testing, combined with the exponential nature of Bayesian inference, the application often ran into memory allocation errors (and out of memory int he full trials). In order to free memory for the application to run, the built-in Java garbage collector is invoked. Java objects are created on heap. The Java garbage collector attempts to free heap space by collecting those objects that are either set to null, whose parents are set to null or that are created within a block for which the scope is already passed in the execution. The reason that the garbage collector adds to the time needed to complete the tests, is that execution is paused each time the collector is invoked.

7 Conclusions

Based on the fact that it supports both polytime model counting as well as polytime conditioning, d-DNNF and sd-DNNF appear to be the theoretical best suited target languages for general mHealth applications that use Bayesian network inference. The fact that the main improvements in these types of applications lies in the improvement of inference speed, the two aforementioned properties should generally be sufficient to increase the applications' performance. As d-DNNF and sd-DNNF are the most succinct languages to support the required query and the required transformation, there should generally be no need to select a different target language. From a practical standpoint, d-DNNF ranks at the top of the candidate languages, due to the fact that a stable compiler from Bayesian networks to d-DNNF is readily available. In practice, compiling a Bayesian network to d-DNNF shows the same promising results as the theory predicts. While inference speed rapidly increases with the size of the network in an uncompiled network, increases in inference time in the compiled network occur at a much lower rate. While the time needed for the compilation process does increase with the network size, this is merely a one time operation. This fact, combined with the fact that large increases in speed in the on-line phase are observed, especially in larger networks, justifies the usage of compilation into d-DNNF form for any mHealth that uses Bayesian network inference.

Future research could focus on compiling a Bayesian network into the newer target language SDD and testing its performance in a real world application. The reason for this is that this language ranks right behind d-DNNF and sd-DNNF in terms of succinctness, but it has an advantage over these languages in terms of the number of transformations that are supported in polynomial time. The main improvement that these additionally supported transformations offer is an increase in compilation speed. For the networks tested, compilation speed was not a limiting factor, however much larger networks might be compilable into SDD in reasonable time whereas this is not the case for d-DNNF. A compiler for SDD already exists, though its performance appears to differ between different inputs. The existence of a compiler for SDD should however increase the feasibility of testing this language in practice.

A second improvement that could be made to the practical implementation of the compiled d-DNNF network is the loading of the network into the mobile phone application. In the application's current form, the network is loaded into the on-line inference engine on application start-up. This process takes up unnecessary time and should ideally be moved to the off-line phase.

In the end, d-DNNF appears to perform well in practical situations when used in mHealth applications that use Bayesian network inference. Though it would be interesting to examine if there are situation in which SDD would be preferred over d-DNNF, the current improvements that have been made compared to the usage uncompiled networks in mHealth applications seem to be a very promising step in the direction of an accelerated healthcare system.

References

[1] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[2] G. F. Cooper, "The computational complexity of probabilistic inference using bayesian belief networks," *Artificial intelligence*, vol. 42, no. 2, pp. 393–405, 1990.

[3] M. Velikova, P. J. Lucas, and M. Spaanderman, "e-momcare: a personalised home-monitoring system for pregnancy disorders," in *Electronic Healthcare*, pp. 267–274, Springer, 2012.

[4] M. Velikova, J. T. van Scheltinga, P. J. Lucas, and M. Spaanderman, "Exploiting causal functional relationships in bayesian network modelling for personalised healthcare," *International Journal of Approximate Reasoning*, 2013.

[5] A. Darwiche and P. Marquis, "A knowledge compilation map," *Journal of Artificial Intelligence Research*, vol. 17, pp. 229–264, 2002.

[6] M. Chavira and A. Darwiche, "On probabilistic inference by weighted model counting," *Artificial Intelligence*, vol. 172, no. 6, pp. 772–799, 2008.

[7] A. Darwiche, "SDD: A new canonical representation of propositional knowledge bases," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, p. 819, 2011.

[8] Y. Xue, A. Choi, and A. Darwiche, "Basing decisions on sentences in decision diagrams," in *Proceedings of the 26th Conference on Artificial Intelligence (AAAI - Association for the Advancement of Artificial Intelligence)*, pp. 842–849, 2012.

[9] G. V. d. Broeck and A. Darwiche, "On the role of canonicity in bottom-up knowledge compilation," *arXiv preprint arXiv:1404.4089*, 2014.

[10] M. Kay, "mhealth: New horizons for health through mobile technologies," *World Health Organization*, 2011.

[11] I. T. Union, "I.T.U. world telecommunication, ICT indicators database," 2012.

[12] P. Jha, E. Kalra, M. Puleio, A. Ghosh, M. Beckman, and V. Jennings, "Cycletel: Expanding access to family planning through mobile phones," *Proceedings of M4D 2012 28-29 February 2012 New Delhi, India*, vol. 28, no. 29, p. 51, 2012.

[13] L. W. Chang, J. Kagaayi, H. Arem, G. Nakigozi, V. Ssempijja, D. Serwadda, T. C. Quinn, R. H. Gray, R. C. Bollinger, and S. J. Reynolds, "Impact of a mhealth intervention for peer health workers on aids care in

rural uganda: a mixed methods evaluation of a cluster-randomized trial," *AIDS and Behavior*, vol. 15, no. 8, pp. 1776–1784, 2011.

[14] V. W. Consulting, "mhealth for development: the opportunity of mobile technology for healthcare in the developing world," *Washington DC and Berkshire, UK*, 2009.

[15] P. Rubel, J. Fayn, L. Simon-Chautemps, H. Atoui, M. Ohlsson, D. Telisson, S. Adami, S. Arod, M. C. Forlini, C. Malossi, *et al.*, "New paradigms in telemedicine: ambient intelligence, wearable, pervasive and personalized," *Stud Health Technol Inform*, vol. 108, pp. 123–132, 2004.

[16] A. Minutolo, G. Sannino, M. Esposito, and G. De Pietro, "A rule-based mhealth system for cardiac monitoring," in *Biomedical Engineering and Sciences (IECBES), 2010 IEEE EMBS Conference on*, pp. 144–149, IEEE, 2010.

[17] K. Pipatsrisawat and A. Darwiche, "New compilation languages based on structured decomposability," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI - Association for the Advancement of Artificial Intelligence)*, pp. 517–522, 2008.

[18] K. Pipatsrisawat and A. Darwiche, "A lower bound on the size of decomposable negation normal form," in *In Proceedings of the Twenty-Forth AAAI Conference on Artificial Intelligence (AAAI - Association for the Advancement of Artificial Intelligence)*, pp. 345–350, 2010.

[19] Y. Breitbart, H. Hunt III, and D. Rosenkrantz, "On the size of binary decision diagrams representing boolean functions," *Theoretical Computer Science*, vol. 145, no. 1, pp. 45–69, 1995.

[20] UCLA, "Ace compiler." `http://reasoning.cs.ucla.edu/ace/`, 2007. [Online].

[21] F. G. Cozman, "Embedded Bayes." `http://www.cs.cmu.edu/~javabayes/EBayes/index.html/`, 1999. [Online].

[22] J. Gehring, "Android GraphView." `http://android-graphview.org/`, 2014. [Online].

[23] A. Darwiche, "On the tractable counting of theory models and its application to truth maintenance and belief revision," *Journal of Applied Non-Classical Logics*, vol. 11, no. 1-2, pp. 11–34, 2001.

[24] J. Gergov and C. Meinel, "Efficient boolean manipulation with OBDDs can be extended to FBDDs," *IEEE Transactions on Computers*, vol. 43, no. 10, pp. 1197–1209, 1994.

[25] R. Drechsler and D. Sieling, "Binary decision diagrams in theory and practice," *International Journal on Software Tools for Technology Transfer*, vol. 3, no. 2, pp. 112–136, 2001.

[26] S. Ponzio, "A lower bound for integer multiplication with read-once branching programs," *SIAM Journal on Computing*, vol. 28, no. 3, pp. 798–815, 1998.

[27] A. C. Powell, A. B. Landman, and D. W. Bates, "In search of a few good apps," *JAMA - Journal of the American Medical Association*, 2014.

[28] A. Morgado and J. Marques-Silva, "Good learning and implicit model enumeration," in *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on*, pp. 6–pp, IEEE, 2005.

[29] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. 100, no. 8, pp. 677–691, 1986.

[30] W. Günther and R. Drechsler, "Minimization of free BDDs," *Integration, the VLSI Journal*, vol. 32, no. 1, pp. 41–59, 2002.