General Yet Computationally Efficient Aggregation Frameworks

Ronald de Haan

Institute for Logic, Language and Computation (ILLC)
University of Amsterdam

me@ronalddehaan.eu
www.ronalddehaan.eu

What do I want to tell you?

General yet computationally efficient aggregation frameworks

- Aggregation of individuals' opinions
- Trade-off between generality and computational efficiency

- *Description of a research direction, with some recent (preliminary) results sprinkled throughout*

Desiderata

- It would be useful to have an aggregation framework:

  (1) that is general enough to model different relevant scenarios

  and that allows freedom to specify additional constraints (that differ per application)

  (2) that provides a wide range of aggregation rules, that satisfy different normative properties

  (if it's general enough, we can't have all properties we want)

  (3) that allows efficient computation of outcomes of these rules

- We need a trade-off between generality and efficiency!

Example 1: Participatory Budgeting

Budget

Projects

Example 1: Participatory Budgeting

- Finite set $P = \{p_1, \ldots, p_m\}$ of projects, each associated with a cost $c_i \in \mathbb{N}$
- Budget $b \in \mathbb{N}$
- Individuals' votes: a subset $P_i \subseteq P$ of projects s.t.:

$$\sum_{p_j \in P_i} c_j \leq b$$

- Collective outcome: a subset $O \subseteq P$ of projects s.t.:

$$\sum_{p_j \in O} c_j \leq b$$

- *Nontrivial setting—e.g., simple majority doesn't always work*

Example 1: additional constraints

- Now suppose further that some authority specifies further constraints on the outcome, e.g.:
  - "At least x percent of the budget needs to be spent on sustainable projects."
  - "For each infrastructure project that is funded, at least one cultural project needs to be funded."

- *Feature request: be able to specify different constraints on individual opinions and collective outcome*

Example 2: committee elections with constraints

- Finite set $A = \{a_1, \ldots, a_m\}$ of alternatives
- Specification of the size $k \in \mathbb{N}$ of the sought committee
- Individuals' votes: a linear order $\succ_i \in \mathcal{L}(A)$
- Collective outcome: a committee $C \subseteq A$ of size $|C| = k$

- Again, consider some external constraints on the outcome, e.g.:
  - "The committee should be gender-balanced."
  - "The expertise of the committee should cover areas A, B, C."

One candidate framework: Judgment Aggregation

- Perhaps existing toolboxes are enough for what we need—perhaps not.
- The first 'usual suspect' to look at: Judgment Aggregation

- *For this talk, we'll stick to judgment aggregation.*
- *In general, we should keep an open mind: Should we extend the framework? Do we want a (new) different framework?*

Judgment Aggregation

- Issues: a set $\mathcal{I} = \{x_1, \ldots, x_n\}$ of propositional variables
- Integrity constraints: propositional 'statements' $\Gamma_{\text{in}}, \Gamma_{\text{out}}$ over the variables $x_1, \ldots, x_n$ (and possibly more variables)
- Ballot: $(b_1, \ldots, b_n) \in \{0,1\}^n$
  - equivalently: truth assignment α to the variables $x_1, \ldots, x_n$
  - input-consistent if α is consistent with Γ_{in}
  - output-consistent if α is consistent with Γ_{out}
- Profile: sequence $\boldsymbol{r} = (r_1, \ldots, r_m)$ of input-consistent ballots
- Judgment aggregation rule: a function F that assigns to each profile $\boldsymbol{r}$ a set $F(\boldsymbol{r})$ of (output-consistent) ballots

Judgment aggregation rules: Kemeny, Slater

- The Kemeny rule selects those output-consistent ballots that minimize the total sum of Hamming distances to the profile:

$$\mathsf{Kemeny}(\boldsymbol{r}) = \operatorname*{arg\,min}_{\substack{r^* \in \{0,1\}^n \\ r^* \models \Gamma_{\mathsf{out}}}} \sum_{r_i \in \boldsymbol{r}} \mathrm{Hamming}(r_i, r^*).$$

- The Slater rule selects those output-consistent ballots that minimize the Hamming distance to the majority outcome:

$$\mathsf{Slater}(\boldsymbol{r}) = \operatorname*{arg\,min}_{\substack{r^* \in \{0,1\}^n \\ r^* \models \Gamma_{\mathsf{out}}}} \mathrm{Hamming}(\mathsf{majority}(\boldsymbol{r}), r^*).$$

Encoding Participatory Budgeting in Judgment Aggregation

- We can express the budget constraints using logic constraints
  - E.g., by building a 'counting circuit',
    or with a formula (using additional variables)
- So Judgment Aggregation is expressive enough to capture (some variant of) Participatory Budgeting
- But: we haven't talked about efficiency yet

Computing outcomes

Decision problem: Outcome(F)

Input:	A set $\mathcal{I}$ of issues, an integrity constraint Γ, a profile $\boldsymbol{r}$, and a partial ballot l.
Question:	Is there some $r^* \in F(\boldsymbol{r})$ such that l agrees with r^*?

- *Keep in mind: we also want to solve other computational tasks, e.g., enumerating outcomes, representing them compactly, reason over them, etc*

Intractability in General

Judgment aggregation rule F	complexity of Outcome(F)
Kemeny	Θ_2^{p}-complete
Slater	Θ_2^{p}-complete
...	Θ_2^{p}-hard (or worse)

The trade-off between generality and efficiency

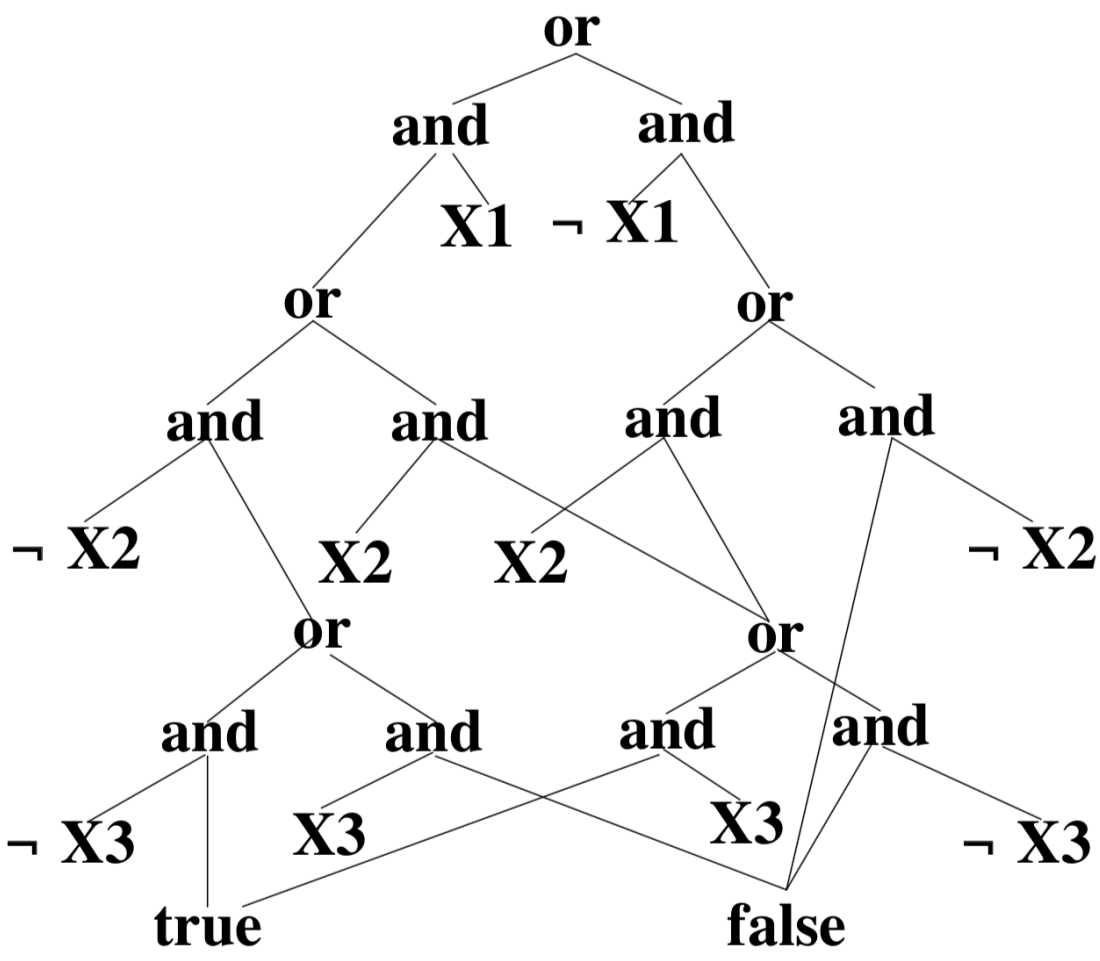
- Rather than take arbitrary logic formulas/circuits as constraints, look at restricted languages that:
  - (i) allow outcomes to be computed efficiently
  - (ii) are still general enough to model interesting settings

- Showcase example:
  - Boolean circuits in DNNF

    (Decomposable Negation Normal Form)

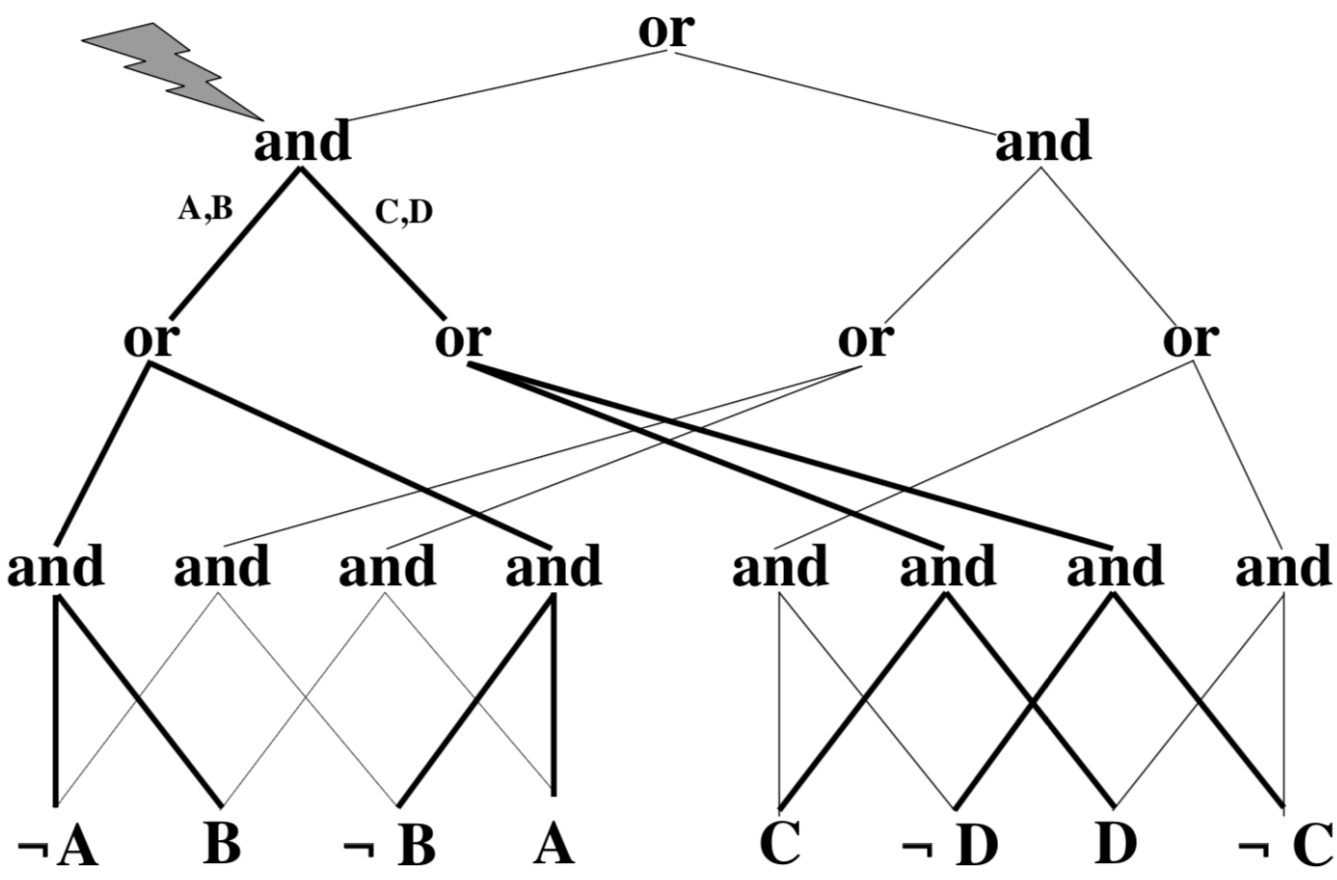
NNF circuits

- Boolean circuits in Negation Normal Form (NNF):



DNNF circuits

- Decomposability: variables inside conjuncts are disjoint



Why are DNNF circuits so fantastic?

- DNNF circuits allow a certain kind of bottom-up computation
- For example:
  - Label literals ℓ with values $\alpha(\ell) \in \mathbb{Z}$,
    such that $\alpha(\ell) + \alpha(\neg\ell) = 0$.
  - Associate $\wedge$-nodes with $+$
    and associate $\vee$-nodes with $\max$.
  - Then computing the value of a DNNF circuit corresponds to computing the value for a DNF formula listing all satisfying truth assignments.

DNNF Circuits

Judgment aggregation rule F	complexity of Outcome(F)
Kemeny	polynomial time
Slater	polynomial time

RdH. Hunting for Tractable Languages for Judgment Aggregation.
Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018).

Encoding Budget Constraints

- A budget constraint $(b, \{c_i\}_{i \in [m]})$ we can translate to a DNNF circuit φ in polynomial time
- We can add to φ in poly time any constraint of the form:
  "Between a and b need to be spent on projects in $P' \subseteq P$"
- We can add to φ in poly time any constraint of the form:
  "At least as many projects in $P_1 \subseteq P$ as in $P_2 \subseteq P$ need to be funded"

- *So judgment aggregation with DNNF circuits strikes a nice balance between generality and efficiency for Participatory Budgeting (with extra constraints)*

Zooming out..

- For some applications, Judgment Aggregation with DNNF constraints is a general yet efficient aggregation framework (for some rules)
  - Also: committee elections with balance constraints, for weakly separable committee scoring rules, can be efficiently encoded in JA/DNNF/Kemeny

---

- *Let's keep looking—what do we want? what can we get?*
  - *What can and can't we encode with DNNF circuits?*
  - *Can we get efficient rules based on proportionality?*
  - *Do we want rankings/scores/etc built-in in the framework?*
  - *(and much more..)*

Relax, this is the last slide.. (or: Summary)

General yet computationally efficient aggregation frameworks

- Aggregation in different scenarios, with constraints
- Trade-off between generality and computational efficiency
- Participatory budgeting, judgment aggregation, DNNF circuits

---

- *Questions for discussion:*
  Do we want to add features to Judgment Aggregation?
  Do we want another framework?