TPLP: Page 1–31. © The Author(s), 2021. Published by Cambridge University Press 2021
doi:10.1017/xxxxx

Unifying Framework for Optimizations in non-boolean Formalisms

YULIYA LIERLER
University of Nebraska Omaha

Abstract

Search-optimization problems are plentiful in scientific and engineering domains. Artificial intelligence has long contributed to the development of search algorithms and declarative programming languages geared towards solving and modeling search-optimization problems. Automated reasoning and knowledge representation are the subfields of AI that are particularly vested in these developments. Many popular automated reasoning paradigms provide users with languages supporting optimization statements. Recall integer linear programming, MaxSAT, optimization satisfiability modulo theory, (constraint) answer set programming. These paradigms vary significantly in their languages in ways they express quality conditions on computed solutions. Here we propose a unifying framework of so called extended weight systems that eliminates syntactic distinctions between paradigms. They allow us to see essential similarities and differences between optimization statements provided by distinct automated reasoning languages. We also study formal properties of the proposed systems that immediately translate into formal properties of paradigms that can be captured within our framework.

1 Introduction

Artificial intelligence is a powerhouse for delivering algorithmic frameworks to support solutions to search-optimization problems that are plentiful in scientific and engineering domains. Automated reasoning and knowledge representation are the subfields of AI that are particularly vested in developing general-purpose search algorithms and declarative programming languages specifically geared towards formulating constraints of search-optimization problems. Various automated reasoning paradigms provide users with languages supporting optimization statements. Indeed, consider such popular paradigms as integer linear programming (ILP) (Papadimitriou and Steiglitz, 1982), MaxSAT (Robinson et al., 2010), optimization satisfiability modulo theory (OMT) (Nieuwenhuis and Oliveras, 2006; Sebastiani and Tomasi, 2012), answer set programming with weak constraints (Alviano, 2018), constraint answer set programming (CASP) (Banbara et al., 2017). These paradigms allow a user to express "hard" and "soft" constraints given a problem of interest. Hard part of an encoding for a considered problem is meant to state requirements on what constitutes a solution to this problem. Soft part of the encoding is meant to state optimization criteria based on which we compare resulting solutions and find optimal ones. For example, integer linear programs have the form

$$\begin{array}{ll} \texttt{maximize} & \mathbf{c^T x} \\ \texttt{subject to} & \mathbf{Ax} \leq \mathbf{b}\,;\ \mathbf{x} \geq 0;\ \texttt{and} \\ & \mathbf{x} \in \mathbb{Z}^n \end{array} \tag{1}$$

where $\mathbf{c}$, $\mathbf{b}$ are vectors and $\mathbf{A}$ is a matrix whose all entries are integers. The `maximize` statement encodes soft constraints, whereas the `subject to` statements encode hard constraints. On the

other hand, in partially weighted MaxSAT the statements to encode both hard and soft part have the form of a clause

$$\ell_1 \vee \cdots \vee \ell_n,$$

where ℓ_i is a literal (recall that a literal is either an atom or its negation, where an atom is a binary/propositional variable). Clauses of the soft part are associated with weights and the goal is to maximize the sum of weights for clauses satisfied by a model of the hard part. These samples of two distinct automated reasoning paradigms offering optimizations point at clear differences. These paradigms vary significantly in their languages, for example, in ways they express the "hard constraints", in ways they express "soft constraints", as well as their vocabularies. Indeed, ILP primarily objects are variables over integers, whereas in MaxSAT we are looking at binary variables. Furthermore, in formalisms such as optimizations modulo theory or constraints answer set programming both kinds of variables are allowed *together with intricate interface between these*.

The relations between many of the enumerated paradigms have been studied in the absence of "soft constraints." Recently, Lierler (2021) provided a formal account comparing MaxSAT/MinSAT family (Robinson et al., 2010) and answer set programs with weak constraints (Alviano, 2018) (weak constraints are syntactic objects to express soft constraints in logic programs). In that work, to draw the precise parallel between these frameworks so called abstract modular weight-systems (w-systems) served the role of a primary tool. These systems abstracted away syntactic differences of the paradigms, while leaving the essential semantic ingredients sufficient to introduce a concept of a model and optimization criteria for their comparison. An abstract notion of a logic introduced by Brewka and Eiter (2007) is a crucial ingredient of w-systems. This abstract logic may encapsulate languages over binary variables. Hence, w-systems may capture frameworks such as MaxSAT or logic programs with optimizations.

In this work, we extend the concepts of an abstract logic and w-systems to provide us with a framework capable to capture formalisms that utilize distinct kinds of variables in their languages. Then, we show how resulting extended w-systems encapsulate ILP, OMT (in its two common variants), and CASP with optimization statements. We trust that such birds eye view on these distinct paradigms will boost cross-fertilization between approaches used in design of algorithms supporting optimizations in distinct fields. Indeed, Lierler (2021; 2022) illustrated how MaxSAT/MinSAT solvers can be used to compute optimal answer sets for logic programs with weak constraints by providing theoretical basis for cross-translations between formalisms. This work provides theoretical grounds for devising translations for related optimization statements in CASP and OMT. Thus, we pave the way, for example, for an extension of a constraint answer set solver EZSMT (Shen and Lierler, 2018). This solver relies on utilizing satisfiability modulo theory solvers for finding answer sets of a considered CASP program. In the future, this system can utilize OMT solvers to find optimal answer sets of a CASP program.

We would like to point at work by Alviano et al. (2018), where the authors also realized the importance of abstracting away the syntactic details of the formalisms to describing hard and soft constraints of a considered problem in order to streamline the utilization of existing efficient solving techniques in new settings. Alviano et al. (2018) define optimization problems at a semantic level to present translations of several preference relations into minimize/maximize and subset/supset statements, as implemented in the ASPRIN system developed by Brewka et al. (2015) and based on answer set programming technology.

The paper is organized as follows. We start the paper by reviewing the concepts of an abstract logic and logic programs. We then define a notion of an extended logic. In Section 3, we show how extended logics naturally capture constraint answer set programs and satisfiability modulo theory formulas (reviewed in the same section). Section 4 introduces the central concept of this work: extended weighted modular systems. Then, we use these modular systems to capture a variety of automated reasoning paradigms with optimization statements, namely, several OMT and CLINGCON-based frameworks. In addition, we provide natural generalizations to these frameworks utilizing introduced modular systems. We conclude by enumerating formal properties of these systems and an account of proofs for presented formal results.

2 Review: Abstract Logic; Logic Programs

A *language* is a set L of *formulas*. A *theory* is a subset of L. Thus the set of theories is closed under union and has the least and the greatest elements: $\emptyset$ and L. This definition ignores any syntactic details behind the concepts of a formula and a theory. A *vocabulary* is an infinite countable set of *atoms*. Subsets of a vocabulary σ represent (classical propositional) *interpretations* of σ. We write $Int(\sigma)$ for the family of all interpretations of a vocabulary σ.

Definition 1
A *logic* is a triple $\mathscr{L} = (L_{\mathscr{L}}, \sigma_{\mathscr{L}}, sem_{\mathscr{L}})$, where

1. $L_{\mathscr{L}}$ is a language (the language of the logic $\mathscr{L}$),
2. $\sigma_{\mathscr{L}}$ is a vocabulary (the vocabulary of the logic $\mathscr{L}$),
3. $sem_{\mathscr{L}} : 2^{L_{\mathscr{L}}} \to 2^{Int(\sigma_{\mathscr{L}})}$ is a function assigning collections of interpretations to theories in $L_{\mathscr{L}}$ (the *semantics* of $\mathscr{L}$).

If a logic $\mathscr{L}$ is clear from the context, we omit the subscript $\mathscr{L}$ from the notation of the language, the vocabulary and the semantics of the logic.

Literals over a vocabulary σ are expressions a and $\neg a$, where a is an atom from σ. A *(logic) rule* over σ is of the form

$$a_0 \leftarrow a_1, \ldots, a_\ell, \; not \; a_{\ell+1}, \ldots, \; not \; a_m, \tag{2}$$

where a_0 is an atom in σ or $\bot$ (empty), and each a_i, $1 \leq i \leq m$, is an atom in σ. A *logic program* over σ is a set of *rules* over σ. The expression a_0 is the *head* of the rule. The expression on the right hand side of the arrow is the *body*. We write $hd(\Pi)$ for the set of nonempty heads of rules in logic program Π. It is customary for a given vocabulary σ, to identify a set X of atoms over σ with (i) the complete and consistent set of literals over σ constructed as $X \cup \{\neg a \mid a \in \sigma \setminus X\}$, and with (ii) an assignment function that assigns the truth value *true* to every atom in X and *false* to every atom in $\sigma \setminus X$. In the sequel, we may refer to sets of atoms as assignments and the other way around following this convention. We say that a set X of atoms *satisfies* rule (2), if X satisfies the propositional formula

$$a_1 \wedge \ldots \wedge a_\ell \wedge \; \neg a_{\ell+1} \wedge \ldots \wedge \; \neg a_m \rightarrow a_0.$$

The *reduct* Π^X of a program Π relative to a set X of atoms is obtained by first removing all rules (2) such that X does not satisfy the propositional formula corresponding to the negative part of the body $\neg a_{\ell+1} \wedge \ldots \wedge \neg a_m$, and replacing all remaining rules with

$$a_0 \leftarrow a_1, \ldots, a_\ell.$$

A set X of atoms is an *answer set*, if it is the minimal set that satisfies all rules of Π^X (Lifschitz et al., 1999). For example, program

$$\begin{array}{l} a \leftarrow not\ b \\ b \leftarrow not\ a. \end{array} \tag{3}$$

has two answer sets $\{a\}$ and $\{b\}$.

A set X of atoms from a vocabulary σ is an *input answer set* of a logic program Π over σ if X is an answer set of the program $\Pi \cup (X \setminus hd(\Pi))$ (Lierler and Truszczynski, 2011). For example, if we consider program (3) as a program over vocabulary $\{a,b,c\}$, then program (3) has four input answer sets: $\{a\}$, $\{b\}$, $\{a,c\}$, $\{b,c\}$.

Brewka and Eiter (2007) showed that their abstract notion of a logic captures default logic, propositional logic, and logic programs under the answer set semantics. For example, the logic $\mathscr{L} = (L, \sigma, sem)$, where

1. L is the set of propositional formulas over σ,
2. $sem(F)$, for a theory $F \subseteq L$, is the set of propositional models of F over σ,

captures propositional logic. We call this logic $\mathscr{L}$ the *pl-logic* and theories (that we later call modules) in the pl-logic, *pl-theories/modules*. If we restrict elements of L to be clauses, then we call $\mathscr{L}$ a *sat-logic*.

Similarly, a logic $\mathscr{L} = (L, \sigma, sem)$, where

1. L is the set of logic program rules over σ,
2. $sem(\Pi)$, for a program $\Pi \subseteq L$, is the set of answer sets/input answer sets of Π over σ,

captures logic programs under the answer set/input answer set semantics. We call these logics the *lp-logic* and *ilp-logic* respectively and theories/modules in these logics, *lp-theories/modules* and *ilp-theories/modules*.

3 Extended Logic for CAS Programs and SMT Formulas

Constraint Satisfaction Problems and Constraint Answer Set Programs A pair $[V,D]$, where V is a set of variables and D is a set of values for variables in V or the *domain* for V, is called a *specification*. A *constraint* over specification $[V,D]$ is a pair $\langle t, \mathbb{R} \rangle$, where t is a tuple of some (possibly all) variables from V and $\mathbb{R}$ is a relation on D of the same arity as t. A collection of constraints over $[V,D]$ is a *constraint satisfaction problem* (CSP) over $[V,D]$. An *evaluation* of V is a function assigning to every variable in V a value from D. An evaluation ν *satisfies* a constraint $\langle (x_1,\ldots,x_n), \mathbb{R} \rangle$ (or is a *solution* of this constraint) if $(\nu(x_1),\ldots,\nu(x_n)) \in \mathbb{R}$. An evaluation *satisfies* (or is a *solution* to) a constraint satisfaction problem if it satisfies every constraint of the problem. Let $c = \langle t, \mathbb{R} \rangle$ be a constraint and D the domain of its variables. Let k denote the arity of t. The constraint $\overline{c} = \langle t, D^k \setminus \mathbb{R} \rangle$ is the *complement* (or *dual*) of c. Clearly, an evaluation of variables in t satisfies c if and only if it does not satisfy $\overline{c}$. Frequently, constraints are stated implicitly without a reference to explicit relation. In particular, constraints over integers or reals are often formulated by means of common arithmetic relation symbols.

Example 1
A constraint $c_1 = \langle x, \langle (1), (2) \rangle \rangle$ over specification $s_1 = [\{x\}, \{0,1,2\}]$ can be implicitly represented as inequality $x \geq 1$ or inequality $x \neq 0$.

When we refer to some *class of constraints* we assume that all of its members are over the same specification. We call a function from a vocabulary σ to a class $\mathscr{C}$ of constraints a $(\sigma, \mathscr{C})$-*denotation*. From now on given a vocabulary σ, we assume that some of its atoms are marked as *irregular/constraint* atoms. We utilize subscripts *r* and *c* to refer to *regular* atoms – atoms not marked as constraint ones – and *constraint* atoms, respectively. Thus, given vocabulary σ: σ_r forms the subset of σ containing all regular atoms and σ_c forms the subset of σ containing all constraint atoms.

We present the definition of answer sets for CAS programs as proposed by Lierler (2014) using the notation of this paper.

Definition 2 (*CAS Programs and their Answer Sets*)
A *CAS rule* over vocabulary σ is a logic rule (2) over σ, where a_0 is an atom in σ_r or $\bot$ (empty), and each a_i, $1 \leq i \leq m$, is an atom in σ. A *constraint answer set program (CAS program) P* over vocabulary σ, class $\mathscr{C}$ of constraints, and $(\sigma_c, \mathscr{C})$-denotation γ is a set of CAS rules over σ. Given a CAS program (or logic program or propositional formula) P, by $At(P)$ we denote atoms occurring in P. A set $X \subseteq At(P)$ is an *answer set* of P if

- $X \cap At(P)_r \subseteq hd(P)$,
- X is an input answer set of P, and
- the following CSP has a solution[1]

$$\{\gamma(a) \mid a \in X \cap At(P)_c\} \cup \{\overline{\gamma(a)} \mid a \in At(P)_c \setminus X\}. \tag{4}$$

A pair (X, ν) is an <u>extended</u> *answer set* of CAS program P if X is an answer set of P and ν is a solution to (4).

Example 2
Consider specification s_1 and a class $\mathscr{C}_1$ of constraints consisting of constraint c_1 and its complement, where s_1 and c_1 are defined in Example 1. Take vocabulary σ_1 contain two regular atoms a, b and an irregular atom $|x \neq 0|$; and assume $(\sigma_{1c}, \mathscr{C}_1)$-denotation that maps irregular atom $|x \neq 0|$ to constraint c_1. Let P_1 be CAS program over σ_1, $\mathscr{C}_1$, and $(\sigma_{1c}, \mathscr{C}_1)$-denotation consisting of rules in (3) and rule

$$\leftarrow a, |x \neq 0|.$$

Take ν_0, ν_1, ν_2 be evaluations assigning x to 0, 1, and 2, respectively. The extended answer sets of P_1 are $(\{a\}, \nu_0)$, $(\{b\}, \nu_0)$, $(\{b, |x \neq 0|\}, \nu_1)$, $(\{b, |x \neq 0|\}, \nu_2)$.

Lierler and Truszczynski (2015) introduced abstract modular systems based of conglomerations of theories over various logics. They illustrated that such systems can be used to capture CAS programs: in particular, they may capture answer sets of a given CAS program. Here, we introduce an extended notion of a logic so that we may speak of abstract systems capturing the meaning of CAS progams in terms of <u>extended</u> answer sets. We then show that the concept of extended logic is helpful in capturing problems expressed as satisfiability modulo theories (SMT).

[1] Gebser et al. (2016) proposed a definition for CAS programs that assumes two kinds of irregular/constraint atoms: strict-irregular and non-strict-irregular. In our presentation here constraint atoms behave as strict-irregular atoms. Changing condition (4) to $\{\gamma(a) \colon a \in X \cap At(\Pi)_c\}$ turns the behavior of constraint atoms to non-strict-irregular atoms.

Extended Logic For a vocabulary σ, set V of variables, and domain D, we call a pair (I, ν), where I is an interpretation over σ and ν is an evaluation from V to D, *an extended interpretation* over σ, V, D. We write $Int(\sigma, V, D)$ for the family of all extended interpretations over σ, V, D.

Definition 3 (*Extended Logic*)
An *extended logic or e-logic* $\mathscr{L}+$ is a tuple

$$(L_{\mathscr{L}+}, \sigma_{\mathscr{L}+}, \Delta_{\mathscr{L}+}, \Upsilon_{\mathscr{L}+}, sem_{\mathscr{L}+}),$$

where

1. $L_{\mathscr{L}+}$ is a language
2. $\sigma_{\mathscr{L}+}$ is a vocabulary
3. $\Delta_{\mathscr{L}+}$ is a domain – a set of values – (the domain of the logic $\mathscr{L}+$)
4. $\Upsilon_{\mathscr{L}+}$ is a a set of variables over domain $\Delta_{\mathscr{L}+}$ (the variables of the logic $\mathscr{L}+$)
5. $sem_{\mathscr{L}+}$: is a function assigning collections of extended interpretations (I, ν) to theories in $L_{\mathscr{L}+}$, where (I, ν) is an element in $Int(\sigma_{\mathscr{L}+}, \Upsilon_{\mathscr{L}+}, \Delta_{\mathscr{L}+})$.

In the sequel, we will default to naming the members of the tuples of extended logic $\mathscr{L}+$ as in this definition.

It is easy to see that an extended logic generalizes the concept of a logic: we can identify any logic $\mathscr{L} = (L_{\mathscr{L}}, \sigma_{\mathscr{L}}, sem_{\mathscr{L}})$ with its extended counterpart $\mathscr{L}+ = (L_{\mathscr{L}}, \sigma_{\mathscr{L}}, \emptyset, \emptyset, sem_{\mathscr{L}})$, where $sem_{\mathscr{L}}$ in $\mathscr{L}+$ is identified with a function that maps theories into pairs whose first element is an interpretation of $sem_{\mathscr{L}}$ in $\mathscr{L}$ application and the second element is empty function.

We now illustrate that extended logic captures CAS programs under extended answer set semantics. Then, we show how SMT formulas are captured by this formalism. Indeed, provided class $\mathscr{C}$ of constraints over specification $[V, D]$ and a $(\sigma_c, \mathscr{C})$-denotation, the extended logic (L, σ, D, V, sem), where

1. L is the set of CAS rules over vocabulary σ;
2. $sem(P)$, for a theory $P \subseteq L$, over class $\mathscr{C}$ of constraints and $(\sigma_c, \mathscr{C})$-denotation, is the set of extended answer sets of P,

captures CAS programs under extended answer set semantics. We call this logic CAS logic.

Satisfiability Modulo Theories as Theories in Extended Logic Here we state the definition of a Satisfiability Modulo Theories (or SMT) formula (Barrett and Tinelli, 2014) following the lines by Lierler and Susman (2017) using terminology introduced in this work. An alternative name for SMT formulas could have been constraint formulas.

Definition 4 (*SMT formulas and their Models*)
An *SMT formula* $\mathscr{F}$ over vocabulary σ, class $\mathscr{C}$ of constraints and $(\sigma_c, \mathscr{C})$-denotation γ is a set of propositional formulas (often assumed to be clauses) over σ. A set $X \subseteq \sigma$ of atoms is a *model* of $\mathscr{F}$, denoted $X \models \mathscr{F}$, if

- X is a model of propositional classical logic theory $\mathscr{F}$, and
- the CSP (4) with P replaced by $\mathscr{F}$ has a solution.

A pair (X, ν) is an *extended model*, denoted $(X, \nu) \models \mathscr{F}$, if X is model of $\mathscr{F}$ and ν is a solution to (4) with P replaced by $\mathscr{F}$.

SMT formulas can be captured by extended logic theories just as CAS programs. Provided class $\mathscr{C}$ of constraints over specification $[V,D]$ and a $(\sigma_c,\mathscr{C})$-denotation, the extended logic (L,σ,D,V,sem), where

1. L is the set of propositional formulas over vocabulary σ;
2. $sem(\mathscr{F})$, for a theory $\mathscr{F}\subseteq L$ over class $\mathscr{C}$ of constraints and $(\sigma_c,\mathscr{C})$-denotation, is the set of extended models of $\mathscr{F}$,

captures SMT formulas. We call this logic SMT-logic (over constraints $\mathscr{C}$). If we restrict elements of L to be conjunctions of literals

$$a_1\wedge\ldots\wedge a_\ell\wedge\ \neg a_{\ell+1}\wedge\ldots\wedge\ \neg a_m. \tag{5}$$

then we call this extended logic a *Restricted SMT or RSMT-logic*.

Example 3
Example 2 defines σ_1, $\mathscr{C}_1$, and $(\sigma_{1c},\mathscr{C}_1)$-denotation considered here. Let clauses

$$\{a\vee b,\ \neg a,\ \neg a\vee\neg|x\neq 0|\}$$

over σ_1, $\mathscr{C}_1$, and $(\sigma_{1c},\mathscr{C}_1)$-denotation form an SMT-logic theory H. Recall how (i) Example 2 is a continuation of Example 1, where domain of variable x was specified as $\{0,1,2\}$ and (ii) valuations ν_0,ν_1,ν_2 are chosen to assign x to 0, 1, 2, respectively. There are three extended interpretations that are extended models to the considered SMT-logic theory H:

$$(\{b\},\nu_0),\ \ (\{b,|x\neq 0|\},\nu_1),\ \ (\{b,|x\neq 0|\},\nu_2).$$

Integer CSP as Extended Logic An *integer expression* has the form

$$b_1x_1+\cdots+b_nx_n,$$

where $b_1,\ldots,b_n$ are integers $\mathbb{Z}$ and $x_1,\ldots,x_n$ are variables over $\mathbb{Z}$. When $b_i=1$ $(1\leq i\leq n)$ we may omit it from the expression. We call a constraint *integer* when it is over specification whose domain is $\mathbb{Z}$ and is encoded implicitly via the form $e\bowtie k$, where e is an integer expression, k is an integer, and $\bowtie$ belongs to $\{<,>,\leq,\geq,=,\neq\}$. We call a CSP an *integer constraint satisfaction problem* when it is composed of integer constraints.

Integer CSPs can be captured by extended logic theories. Indeed, an extended logic $(L,\emptyset,\mathbb{Z},V,sem)$, where

1. L is the set of integer constraints over $[V,\mathbb{Z}]$;
2. $sem(\mathscr{F})$, for a theory $\mathscr{F}\subseteq L$, is the set of pairs $(\emptyset,\nu)$, where evaluation ν is a solution to $\mathscr{F}$,

captures integer CSPs. We call this logic an *I-CSP-logic*. If the domain of this extended logic is that of nonnegative integers $\mathbb{Z}^+$ then we call this logic a *nonnegative I-CSP-logic*.

4 Extended Weighted Abstract Modular Systems or EW-Systems

Lierler and Truszczynski (2015) propose (model-based) abstract modular systems or AMS that allow us to construct heterogeneous systems based of "modules" stemming from a variety of logics. We now generalize their framework by incorporating the notion of an extended logic.

Definition 5 (*Extended Modules (or e-modules) and their Models*)
Let $\mathscr{L}+$ be an extended logic. A theory of $\mathscr{L}+$, that is, a subset of the language $L_{\mathscr{L}+}$ is called an *extended (model-based) $\mathscr{L}+$-module* (or an *e-module*, if the explicit reference to its logic is not necessary).

Let T be an extended $\mathscr{L}+$-module. An extended interpretation (I,ν) over $\sigma_{\mathscr{L}+}$, $\Delta_{\mathscr{L}+}$, $\Upsilon_{\mathscr{L}+}$ is an *extended model* of T, whereas I is a *model* of T if $(I,\nu) \in sem_{\mathscr{L}+}(T)$.

By L_T, σ_T, Δ_T, Υ_T, and sem_T we refer to the elements $L_{\mathscr{L}+}, \sigma_{\mathscr{L}+}, \Delta_{\mathscr{L}+}, \Upsilon_{\mathscr{L}+}$, and $sem_{\mathscr{L}+}$ of module T logic $\mathscr{L}+$, respectively.

As before, we use words theory and modules interchangeably. Furthermore, for a theory/module in SMT-logic we often refer to these as SMT formulas. For a theory/module in CAS-logic we refer to it as a CAS program.

For an interpretation I, by $I_{|\sigma}$ we denote an interpretation over vocabulary σ constructed from I by dropping all its members not in σ. For a set V of variables and an evaluation ν defined over some superset of V, by $\nu_{|V}$ we denote an evaluation over V constructed from ν so that $\nu_{|V}(v) = \nu(v)$ for any variable v in V. For example, let V be the set of variables $\{x,y\}$, and evaluation ν defined over $\{x,y,z\}$ assigns x and y to 1 and variable z to 2. Evaluation $\nu_{|V}$ is defined on domain V and assigns both of its variables value 1.

We now generalize the notion of an extended model to vocabularies and evaluations that go beyond the one of a considered module in a straight forward manner. For an e-module T and an extended interpretation (I,ν) over σ, V, D so that $\sigma_T \subseteq \sigma$, $\Upsilon_T \subseteq V$, $\Delta_T \subseteq D$, we say that (I,ν) is an *extended model* of T, denoted $(I,\nu) \models T$, if $(I_{|\sigma_T}, \nu_{|\Upsilon_T}) \in sem_T$. We can generalize the concept of a model in a similar way.

We call extended logics (L,σ,D,V,sem) and (L',σ',D',V',sem') (and, respectively e-modules in these logics) *coherent* if $D = D'$, whenever $V \cap V' \neq \emptyset$. In other words if e-logics are coherent and they share variables then the domains of these e-logics coincide.

Definition 6 (*Extended Abstract Modular Systems (EAMSs) and their models*)
A set of coherent e-modules, possibly in different logics, over different vocabularies and/or variables, is an *extended (model-based) abstract modular system (EAMS)*. For an extended abstract modular system $\mathscr{H}$,

- the union of the vocabularies of the logics of the modules in $\mathscr{H}$ forms the *vocabulary* of $\mathscr{H}$, denoted by $\sigma_{\mathscr{H}}$,
- the union of the variables of the logics of the modules in $\mathscr{H}$ forms the *set of variables* of $\mathscr{H}$, denoted by $\Upsilon_{\mathscr{H}}$,
- the union of the domains of the logics of the modules in $\mathscr{H}$ forms the *domain* of $\mathscr{H}$, denoted by $\Delta_{\mathscr{H}}$.

An extended interpretation (I,ν) over $\sigma_{\mathscr{H}}$, $\Upsilon_{\mathscr{H}}$, $\Delta_{\mathscr{H}}$ is an *extended model* of $\mathscr{H}$ whereas I is a *model* of $\mathscr{H}$, when for every module $B \in \mathscr{H}$, (I,ν) is an extended model of B.

When an EAMS consists of a single module $\{F\}$ we identify it with module F itself. Just as the concept of an extended logic presented here is a generalization of logic by Brewka and Eiter, the concept of EAMS is a generalization of AMS by Lierler and Truszczynski (2015).

Extended W-systems In practice, we are frequently interested not only in identifying models of a given logical formulation of a problem (hard fragment), but also identifying models that are

deemed optimal according to some criteria (soft fragment). Frequently, multi-level optimizations are of interest. Lierler and Truszczynski (2015) argued how AMS and, consequently, EAMS as its generalization are geared towards capturing heterogeneous solutions for formulating hard constraints. Lierler (2021) used abstract modular systems to formulate a concept of w-systems that enable soft constraints. W-systems are adequate to capture the MaxSAT/MinSAT family of problems (Robinson et al., 2010) as well as logic programs with weak constraints (Alviano, 2018). W-systems provide us with means of studying these distinct logic frameworks under a unified viewpoint.

Definition 7 (*Ew-conditions and their models*)
An *ew-condition* in an extended logic $\mathscr{L}+$ is a pair $(T, w; \mathbf{c}@\ell)$ — consisting of an $\mathscr{L}+$ e-module T and an expression $w; \mathbf{c}@\ell$, where

- w is an integer,
- $\mathbf{c}$ is a function from variables in Υ_T to reals, and
- l is a positive integer.

We refer to integers l and w as *levels* and *weights*, respectively. We refer to function $\mathbf{c}$ as *coefficients* function.

Let B be an ew-condition $(T, w; \mathbf{c}@\ell)$. Intuitively, by σ_B, Δ_B, and Υ_B we refer to σ_T, Δ_T, and Υ_T, respectively. Let $(I, \nu) \in Int(\sigma_T, \Upsilon_T, \Delta_T)$ be an extended interpretation; it is an *extended model* of B, denoted $(I, \nu) \models B$, when (I, ν) is an extended model of T, also in this case I is called a *model*, denoted $I \models B$.

The notion of an (extended) model for ew-condition is generalized to vocabularies and evaluations that go beyond the one of a considered ew-condition in a straight forward manner (as it was done earlier for the case of e-modules). For an extended interpretation (I, ν) over σ, V, D so that $\sigma_B \subseteq \sigma$, $\Upsilon_B \subseteq V$, $\Delta_B \subseteq D$, we say that (I, ν) is an *extended model* of ew-condition B, denoted $(I, \nu) \models B$, if $(I_{|\sigma_T}, \nu_{|\Upsilon_T}) \models B$. The concept of a model is generalized in a similar way.

Intuitively, the role of weights w in ew-condition $B = (T, w; \mathbf{c}@\ell)$ is to distinguish the quality of models/extended models of B given their propositional part; the role of the coefficient $\mathbf{c}$ is to distinguish the quality of extended models of B given their evaluation part. These intuitions become more apparent in the next definition, where we associate "cost" expressions with the models of ew-conditions. We elaborate more on these expressions after their definitions.

Definition 8 (*Cost expressions for (extended) interpretations of ew-conditions*)
Ler B be an ew-condition.

For an extended interpretation (I, ν) over σ, V, D so that $\sigma_B \subseteq \sigma$, $\Upsilon_B \subseteq V$, $\Delta_B \subseteq D$, a mapping $[(I, \nu) \models B]$ is defined as

$$[(I, \nu) \models B] = \begin{cases} \sum\limits_{x \in \Upsilon_B} \nu(x) \cdot \mathbf{c}(x) & \text{when } (I, \nu) \models B \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

For an interpretation I over σ so that $\sigma_B \subseteq \sigma$, a mapping $[I \models B]$ is defined as follows

$$[I \models B] = \begin{cases} w & \text{when } I \models B, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

We view expressions $[(I,\nu) \models B]$ and $[I \models B]$ as costs associated with two distinct parts of an (extended) interpretation of a considered ew-condition B. Indeed, the former expression accounts for *"the quality" of an evaluation* associated with an extended interpretation and utilizes the coefficient of ew-condition B to compute that. The later expression accounts for *"the quality" of "logical/propositional part"* of an interpretation by considering the weight of ew-condition B. It is easy to see that non-zero values for costs are associated only with (extended) interpretations that are also models. These cost expressions are used in formulating the definitions of optimal (extended) models of ew-systems defined next. These ew-systems take ew-conditions as the tool for distinguishing quality of their (extended) models. Formulas (6) and (7) are used within expressions (8) and (10) utilized in the definitions of optimal models and optimal extended models of ew-systems, respectively. In the sequel, Examples 4, 5, 6 illustrate the use of instances of cost expressions (6) and (7) within formulas (8) and (10) .

Before introducing the key concept of this paper – ew-systems – we present a number of useful abbreviations. We identify ew-conditions of the form

- $(T,w;\mathbf{c}@1)$ with expressions $(T,w;\mathbf{c})$: i.e., when the level is missing it is considered to be 1.
- $(T,w;\emptyset@\ell)$ with expressions $(T,w@\ell)$: i.e., when the coefficients function is empty.
- $(T,w;\mathbf{c}@\ell)$ with expressions $(T,w@\ell)$, when the coefficients function $\mathbf{c}$ assigns 0 to every element in its domain.

For example, (T,w) stands for ew-condition, whose level is 1 and whose coefficients function is either empty or assigns 0 to every element in its domain.

For a collection $\mathscr{Z}$ of ew-conditions, its *vocabulary*, denoted by $\sigma_{\mathscr{Z}}$, its *set of variables*, denoted by $\Upsilon_{\mathscr{Z}}$, its *domain*, denoted by $\Delta_{\mathscr{Z}}$ is defined following the lines of these concepts for EAMS. We say that ew-condition $(T,w;\mathbf{c}@\ell)$ is *coherent* with an EAMS $\mathscr{H}$ if

- $\sigma_T \subseteq \sigma_{\mathscr{H}}$,
- $\Upsilon_T \subseteq \Upsilon_{\mathscr{H}}$,
- given any module H in $\mathscr{H}$, $\Delta_T = \Delta_H$ whenever $\Upsilon_T \cap \Upsilon_H \neq \emptyset$.

Definition 9 (*Ew-systems and their models*)
A pair $(\mathscr{H},\mathscr{Z})$ consisting of an EAMS $\mathscr{H}$ and a set $\mathscr{Z}$ of ew-conditions so that every element in $\mathscr{Z}$ is coherent with $\mathscr{H}$ is called an *ew-system* ($\mathscr{H}$ and $\mathscr{Z}$ intuitively stand for *hard* and *soft*, respectively).

Let $\mathscr{W} = (\mathscr{H},\mathscr{Z})$ be an ew-system. The vocabulary of $\mathscr{H}$ forms the *vocabulary* of $\mathscr{W}$, denoted by $\sigma_{\mathscr{W}}$. Similarly, $\Upsilon_{\mathscr{W}} = \Upsilon_{\mathscr{H}}$ and $\Delta_{\mathscr{W}} = \Delta_{\mathscr{H}}$. An extended interpretation (I,ν) in $Int(\sigma_{\mathscr{W}},\Upsilon_{\mathscr{W}},\Delta_{\mathscr{W}})$ is an *extended model* of $\mathscr{W}$, whereas I is a *model*, if (I,ν) is an extended model of $\mathscr{H}$.

For a level l, by $\mathscr{W}_l$ we denote the subset of $\mathscr{Z}$ that includes all ew-conditions whose level is l. By $\lambda(\mathscr{W})$, we denote the set $\{l \mid (T,w;\mathbf{c}@\ell) \in \mathscr{Z}\}$ of all levels associated with ew-system $\mathscr{W}$. For a level $l \in \lambda(\mathscr{W})$ by $l^{\uparrow}$ we denote the least level in $\lambda(\mathscr{W})$ that is greater than l (it is obvious that for the greatest level in $\lambda(\mathscr{W})$, $l^{\uparrow}$ is undefined). For example, for levels in $\{2,6,8,9\}$, $2^{\uparrow} = 6$, $6^{\uparrow} = 8$, and $8^{\uparrow} = 9$.

Definition 10 (*Optimal models of ew-systems*)
For level $l \in \lambda(\mathscr{W})$, a model I^* of ew-system $\mathscr{W}$ is *l-optimal* if I^* satisfies equation

$$I^* = arg\max_{I} \sum_{B \in \mathscr{W}_l} [I \models B], \tag{8}$$

where

- I ranges over models of $\mathscr{W}$ if l is the greatest level in $\lambda(\mathscr{W})$,
- I ranges over $l^{\uparrow}$-optimal models of $\mathscr{W}$, otherwise.

We call a model *l-min-optimal* if max is replaced by min in (8) (and occurrences of word *optimal* are replaced by *min-optimal* in the definition; we drop this remark from the later similar definitions).

A model I^* of $\mathscr{W}$ is *optimal* if I^* is l-optimal model for every level $l \in \lambda(\mathscr{W})$. A model I^* of $\mathscr{W}$ is *min-optimal* if I^* is l-min-optimal model for every level $l \in \lambda(\mathscr{W})$.

We now provide intuitions for sub-expression

$$\sum_{B \in \mathscr{W}_l} [I \models B] \tag{9}$$

of the formula (8). Given an interpretation I, formula (9) can be seen as a cost of this interpretation with respect to level l of ew-system $\mathscr{W}$. This cost is computed by summing all the weights of the ew-conditions of $\mathscr{W}_l$ for which this interpretation I is a model. When ew-system $\mathscr{W}$ contains only ew-conditions of a single level then formula (9) equips us with the cost of the considered interpretation with respect to the overall system.

Definition 11 (*Extended Optimal Models of Ew-systems*)
For level $l \in \lambda(\mathscr{W})$, an extended model (I^*, ν^*) of ew-system $\mathscr{W}$ is *l-optimal* if (I^*, ν^*) satisfies equation

$$(I^*, \nu^*) = arg\max_{(I^*, \nu^*)} \sum_{B \in \mathscr{W}_l} ([I \models B] + [(I, \nu) \models B]), \tag{10}$$

where

- (I, ν) ranges over extended models of $\mathscr{W}$ if l is the greatest level in $\lambda(\mathscr{W})$,
- (I, ν) ranges over $l^{\uparrow}$-optimal extended models of $\mathscr{W}$, otherwise.

We call a model *l-min-optimal* if max is replaced by min in the equation above.

An extended model (I^*, ν^*) of $\mathscr{W}$ is *optimal* if (I^*, ν^*) is l-optimal model for every level l in $\lambda(\mathscr{W})$. An extended model (I^*, ν^*) of $\mathscr{W}$ is *min-optimal* if (I^*, ν^*) is l-min-optimal model for every level l in $\lambda(\mathscr{W})$.

We now provide intuitions for sub-expression

$$\sum_{B \in \mathscr{W}_l} ([I \models B] + [(I, \nu) \models B]) \tag{11}$$

of the formula (10). Given an extended interpretation (I, ν), formula (11) can be seen as a cost of this interpretation with respect to level l of ew-system $\mathscr{W}$. This cost is computed by considering all the ew-conditions of $\mathscr{W}_l$ for which this extended interpretation is a model and summing all their weights together with the values provided by linear expression formed within cost expression provided at the first line of (6). When ew-system $\mathscr{W}$ contains only ew-conditions of a single

level then formula (11) equips us with the cost of the considered extended interpretation with respect to the overall system.

If we compare the notion of a optimal/min-optimal model versus a optimal/min-optimal extended model then the former does not take into account the numeric values associated with evaluations corresponding to extended models associated with the considered model; whereas the latter combines the quality of both parts of extended model.

Lierler (2022) noted that the definition of optimal models in terms of "*arg* max"-equation comes from the traditions of literature related to MaxSAT problem. In answer set (logic) programming community, the conditions on optimality of answer sets is stated in terms of "domination" relation. Here we follow the steps by Lierler (2022) and provide an alternative definition to optimal models of ew-systems in terms of domination.

Definition 12 (*Optimal models of ew-systems*)
Let I and I' be models of ew-system $\mathscr{W}$. Model I' *min-dominates* I if there exists a level $l \in \lambda(\mathscr{W})$ such that following conditions are satisfied:

1. for any level $l' > l$ the following equality holds

$$\sum_{B \in \mathscr{W}_{l'}} [I \models B] = \sum_{B \in \mathscr{W}_{l'}} [I' \models B]$$

2. the following inequality holds for level l

$$\sum_{B \in \mathscr{W}_{l}} [I' \models B] < \sum_{B \in \mathscr{W}_{l}} [I \models B]$$

Model I' *max-dominates* I if we change less-than symbol by greater-than symbol in the inequality of Condition 2.

A model I^* of $\mathscr{W}$ is *optimal* if there is no model I' of $\mathscr{W}$ that max-dominates I^*. A model I^* of $\mathscr{W}$ is *min-optimal* if there is no model I' of $\mathscr{W}$ that min-dominates I^*.

Definition 13 (*Optimal extended models of ew-systems*)
Let (I, ν) and (I', ν') be extended models of ew-system $\mathscr{W}$. Extended model (I', ν') *min-dominates* (I, ν) if there exists a level $l \in \lambda(\mathscr{W})$ such that following conditions are satisfied:

1. for any level $l' > l$ the following equality holds

$$\sum_{B \in \mathscr{W}_{l'}} ([I \models B] + [(I, \nu) \models B]) = \sum_{B \in \mathscr{W}_{l'}} ([I' \models B] + [(I', \nu') \models B])$$

2. the following inequality holds for level l

$$\sum_{B \in \mathscr{W}_{l}} ([I' \models B] + [(I', \nu') \models B]) < \sum_{B \in \mathscr{W}_{l}} ([I \models B] + [(I, \nu) \models B])$$

Extended model (I', ν') *max-dominates* (I, ν) if we change less-than symbol by greater-than symbol in the inequality of Condition 2.

An extended model (I^*, ν^*) of $\mathscr{W}$ is *optimal* if there is no extended model (I', ν') of $\mathscr{W}$ that max-dominates (I^*, ν^*). An extended model (I^*, ν^*) of $\mathscr{W}$ is *min-optimal* if there is no extended model (I', ν') of $\mathscr{W}$ that min-dominates (I^*, ν^*).

Proposition 1
Definitions 10 and 12 are equivalent. Definitions 11 and 13 are equivalent.

5 Instances of Ew-Systems

5.1 MaxSMT-family and Optimization Modulo Theories

Lierler (2021; 2022) illustrated the utility of w-systems — a precursor of ew-systems — by using these to capture the definitions of *MaxSAT*, *weighted MaxSAT*, and *partially weighted MaxSAT* (or, *pw-MaxSAT*) (Robinson et al., 2010). Here we look into capturing *weighted MaxSMT* and *pw-MaxSMT*.

We start by introducing some useful notation. For a vocabulary σ, a specification $[V,D]$, and an e-logic $\mathscr{L}+$ so that $\sigma_{\mathscr{L}+}=\sigma$, $\Delta_{\mathscr{L}+}=D$, $\Upsilon_{\mathscr{L}+}=V$, an e-module $T_{\mathscr{L}+}$ is called *σ,V,D-theory/σ,V,D-module* when $sem(T_{\mathscr{L}+})=Int(\sigma,V,D)$. In other words, any extended interpretation in $Int(\sigma,V,D)$ is an extended model of a σ,V,D-theory, and, consequently, any interpretation in $Int(\sigma)$ is a model of a σ,V,D-theory. Note how specific language of e-logic of a σ,V,D-theory becomes immaterial. Thus, we allow ourselves to denote an arbitrary σ,V,D-theory by $T_{\sigma,V,D}$ disregarding the reference to its e-logic. Also recall that the (extended) interpretations/models of any theory are generalized to signatures that go beyond original signature of the considered theory.

Definition 14 (*Weighted MaxSMT problems and their solutions*)
A *weighted MaxSMT problem* (Nieuwenhuis and Oliveras, 2006) over vocabulary σ, class $\mathscr{C}$ of constraints and $(\sigma_c,\mathscr{C})$-denotation is defined as a set S of pairs $(\mathscr{F},w)$, where $\mathscr{F}$ is an SMT formula over σ, $\mathscr{C}$, and $(\sigma_c,\mathscr{C})$-denotation so that $\mathscr{F}$ is a clause, and w is a positive integer[2]. An interpretation $I^*\in Int(\sigma)$ is a *solution* to weighted MaxSMT problem S, when it satisfies the equation

$$I^* = \arg\max_{I} \sum_{(\mathscr{F},w)\in S} w\cdot[I\models\mathscr{F}], \text{ where}$$

$$[I\models\mathscr{F}] = \begin{cases} 1 & \text{when } I\models\mathscr{F} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

and I ranges over all interpretations in $Int(\sigma)$.

Example 4
Consider specification s_1, class $\mathscr{C}_1$ of constraints consisting of c_1 and its complement, vocabulary σ_1, and $(\sigma_{1c},\mathscr{C}_1)$-denotation from Example 2. Set

$$\{(a\vee b,2),(\neg a,3),(\neg a\vee\neg|x\neq 0|,1)\}$$

exemplifies a weighted MaxSMT problem. Its solutions are $\{b\}$ and $\{b,|x\neq 0|\}$.

Proposition 2
Let S be a weighted MaxSMT problem. The optimal models of ew-system $(T_{\sigma_S,\Upsilon_S,\Delta_S},S)$ — where each element in S is understood as an ew-condition, whose theory is in SMT-logic — form the set of solutions for weighted MaxSMT problem S.

Proposition 2 allows us to view ew-systems as a mean to state definitions of semantics of various formalisms captured by e-logics with optimizations in a uniform way. Indeed, we can formulate a definition of semantics of weighted MaxSMT problem by means of its relation to a respective

[2] Nieuwenhuis and Oliveras (2006) allow w to be a positive real number.

ew-system: *For a weighted MaxSMT problem S over vocabulary σ, class $\mathscr{C}$ of constraints and $(\sigma_c, \mathscr{C})$-denotation, an optimal model of ew-system $(T_{\sigma_S, \Upsilon_S, \Delta_S}, S)$ is a* solution *to S.* In the sequel, we often take this approach to present the definitions of various optimization formalisms and their possible extensions. This allows us to bypass the complexity of varying terminology and notation coming from papers that introduce these formalisms. It is also worth to reiterate that we speak of papers stemming from various research communities that have their own traditions.

Note how in ew-system $(T_{\sigma_S, \Upsilon_S, \Delta_S}, S)$, where S is a weighted MaxSMT problem,

- levels of all ew-conditions are identical (set to 1) and
- the coefficients function assigns all variables to 0.

These observations lend themselves to a natural generalization of weighted MaxSMT problem.

Definition 15 (*Generalized weighted MaxSMT problems and their (extended) solutions*)
A *generalized weighted MaxSMT problem* over class $\mathscr{C}$ of constraints is defined as a set S of ew-conditions in SMT-logic over $\mathscr{C}$. Optimal models and optimal extended models of ew-system $(T_{\sigma_S, \Upsilon_S, \Delta_S}, S)$ form *solutions* and *extended solutions* of S.

Note how extended solutions of S take into account the quality of evaluation that is part of extended model. Also, levels become naturally incorporated into the framework.

Example 5
Let us continue Example 4. Let f_x be a coefficients function defined as $f_x(0) = 10$, $f_x(1) = 100$, $f_x(2) = 1000$. The following set $\{(a \vee b, 2), (\neg a, 3), (\neg a \vee \neg |x \neq 0|, 1; f_x)\}$ over σ_1, $\mathscr{C}_1$, and $(\sigma_{1c}, \mathscr{C}_1)$-denotation forms a generalized weighted MaxSMT problem. Its solutions are $\{b\}$ and $\{b, |x \neq 0|\}$. Its extended solution is $(\{b, |x \neq 0|\}, \nu_2)$, where ν_2 is introduced in Example 2. If we redefine f_x as follows $f_x(0) = 1000$, $f_x(1) = 100$, $f_x(2) = 10$, then the extended solution to considered problem is $(\{b\}, \nu_0)$.

We now define/generalize partially weighted MaxSMT (pw-MaxSMT) problem inspired by the notion of partially weighted MaxSAT problem (Robinson et al., 2010).

Definition 16 (*Generalized partially weighted MaxSMT problems and their (extended) solutions*)

A *generalized partially weighted MaxSMT (or gpw-MaxSMT) problem* over vocabulary σ, class $\mathscr{C}$ of constraints over specification $[V, D]$ and $(\sigma_c, \mathscr{C})$-denotation is defined as ew-system (H, S), where H is a theory in SMT-logic over $\mathscr{C}$ and S is a collection of ew-conditions whose theories are in SMT-logic over $\mathscr{C}$. Formula H is referred to as *hard* problem fragment, whereas S forms *soft* problem fragment. Optimal models and optimal extended models of ew-system (H, S) form *solutions* and *extended solutions* of (H, S).

5.1.1 νZ approach

As mentioned in the introduction, often the complete spectrum of capabilities of automated reasoning systems is described in papers and tutorials by means of examples appealing to intuitions of the readers. We argue that the presented framework provides convenient means to make such presentations formal. For example, Bjørner et al. (2015), the authors of SMT solver νZ, state that

> νZ extends the functionality of Z3 (de Moura and Bjørner, 2008) to include optimization objectives. It allows users to solve SMT constraints and at the same time formulate optimality criteria for the solutions. ...

They then specify that νZ adds to SMT-LIB (Barrett et al., 2016; Barrett et al., 2010) – standard descriptions/language of background theories used in SMT systems – a command of the form

```
(assert-soft F [:weight n]).
```

This command is said to assert soft constraint F, optionally with an integer weight n.[3] If no weight is given, the default weight is 1. Code similar to the one presented in LHS of Figure 1 and the respective output of system νZ presented in RHS of the figure are used by Bjørner et al. (2015) to illustrate a use of soft constraints.

```
LHS:
(declare-fun x () Int)
(declare-fun y () Int)
(define-fun a1 () Bool (> x 0))
(define-fun a2 () Bool (< x y))
(assert (=> a2 a1))
(assert-soft a2 :weight 3)
(assert-soft (not a1) :weight 5)
(check-sat)
(get-model)
```

```
RHS:
sat
 ((define-fun a1 () Bool
    (not (<= x 0)))
  (define-fun x () Int
    0)
  (define-fun a2 () Bool
    (not (<= y x)))
  (define-fun y () Int
    0))
```

Fig. 1: LHS: SMT-LIB code suggesting to maximize $3 \cdot a2 + 5 \cdot \overline{a1}$. RHS: νZ finds a solution where $x = y = 0$.

We now elaborate on this example by Bjørner et al. (2015) to uncover its implicit assumptions about readers intuitions and interpretations of the seen code and its behaviour. Given code in Figure 1 (LHS) as an input, system z3-4.8.12 produces a model where x and y are both assigned to 0 suggesting that $a1$ and $a2$ are assigned to $false$. The authors of νZ informally state that code in Figure 1 (LHS) maximizes expression

$$3 \cdot a2 + 5 \cdot \overline{a1}. \tag{13}$$

To make this claim precise we recall that system νZ looks for models of a provided SMT formula specified in its `assert` statements — an implication $a2 \rightarrow a1$ in this running example, where $a1$ and $a2$ are constraint atoms associated with (via `define-fun` statements) integer constraints $x > 0$ and $x < y$, respectively. Expressions $a2$ and $\overline{a1}$ in (13) should be interpreted in relation with some model X for the encoded SMT formula so that, for instance,

- $a2$ is mapped to 0 if model X assigns $a2$ to $false$, and 1 otherwise; similarly
- $\overline{a1}$ is mapped to 1 if model X assigns $a1$ to $false$, and 0 otherwise.

It is easy to see that there are three models to the implication $a2 \rightarrow a1$. Furthermore, for each of these three models there is an evaluation that maps integer variables x and y into values that satisfy integer constraints associated with constraint atoms $a1$ and $a2$. Thus, all these models form solutions to the SMT formula specified in Figure 1. These models together with the respective

[3] The authors also allow a keyword for decimal weights, yet at the time of writing this paper the system z3-4.8.12, the latest available incarnation of z3 with the functionality of νZ gave out an error message "invalid keyword argument" suggesting that this feature is no longer supported.

value of expression (13) follow:

$a1$	$a2$	$3 \cdot a2 + 5 \cdot \overline{a1}$
$false$	$false$	5
$true$	$true$	3
$true$	$false$	0

It is easy to see that the model listed first maximizes value of expression (13) in comparison to other models.

Generalized partially weighted MaxSMT problems and their solutions can be seen as formal specification of the language supported by the SMT solver νZ. Indeed, the statements that follow the key word `assert` correspond to hard problem fragment; whereas the statements that follow the key word `assert-soft` form soft problem fragment. In other words, we view νZ code as a specification of a gpw-MaxSMT problem, whereas we consider system νZ to compute solutions for given gpw-MaxSMT problems. Let us go back to the νZ code in Figure 1. It specifies the gpw-MaxSMT problem over vocabulary $\{a1, a2\}$, class of integer constraints over specification $[\{x, y\}, \mathbb{Z}]$, and denotation that maps $a1$ and $a2$ into integer constraints $x > 0$ and $x < y$, respectively; so that hard problem fragment consists of propositional formula $a2 \rightarrow a1$, and soft problem fragment consists of ew-conditions $(a2, 3)$ and $(\neg a1, 5)$. It is easy to see that interpretation mapping $a1$ and $a2$ to $false$ forms a solution to the constructed gpw-MaxSMT problem.

Above, we complemented an earlier description of νZ input-output provided by an example with formal specification of these entities. It is also due to note that gpw-MaxSMT problems are more general than SMT-LIB specifications supported by νZ. For example, the notion of a level is present in gpw-MaxSMT specifications of its soft fragment. Also, the notion of an extended solution is defined for gpw-MaxSMT problems. This may provide inspirations for possible extensions to system νZ.

5.1.2 Cost-variable approach

Sebastiani and Tomasi (2012) observed that MaxSMT and its variants encapsulated by gpw-MaxSMT when (non-extended) solutions are considered support optimizations that focus on "propositional part" of a problem encoded within SMT framework. They then proposed an alternative approach that ranks extended models of an SMT formula by value of one of its variables occurring in constraints associated with an SMT formula disregarding any information of a model from a propositional side. In particular, the value of that variable is to be minimized.

We now use ew-systems to capture an optimization satisfiability module problem (OMT) by Sebastiani and Tomasi (2012). To proceed to the definition recall the notion of a σ, V, D-theory, denoted $T_{\sigma,V,D}$, introduced in the beginning of Section 5.1.

Definition 17 (*OMT problems and their solutions*)
An *OMT problem* over vocabulary σ, class $\mathscr{C}$ of constraints over specification $[V, D]$ and $(\sigma_c, \mathscr{C})$-denotation and variable v occurring in V is defined as ew-system (H, S), where H is a theory in SMT-logic over σ, $\mathscr{C}$, and $(\sigma_c, \mathscr{C})$-denotation and S consists of a single ew-condition of the form $(T_{\emptyset,\{v\},D}, 0; (v) \mapsto 1)$; We call min-optimal extended models of this ew-systems *solutions to OMT*.

The use of ew-condition $(T_{\emptyset,\{v\},D}, 0; (v) \mapsto 1)$ in this definition as the only criterion for optimization captures the idea of OMT that relies on the choice of a single variable in constraints to be

minimized. Indeed, the first component $T_{\emptyset,\{v\},D}$ of this ew-condition is such that any (extended) interpretation of theory H in SMT-logic with variable v occurring in it is a model of $T_{\emptyset,\{v\},D}$. The cost of any extended interpretation will be identified with the value of x assigned by this interpretation based on formula (11) used in the definition of the min-optimal extended model (for this instance, $\mathscr{W}_l$ in (11) is a singleton composed of $T_{\emptyset,\{v\},D}$). Indeed, take (I,ν) be an arbitrary model of $B = T_{\emptyset,\{v\},D}$, then $[I \models B] = 0$ and $[(I,\nu) \models B] = \nu(x)$; the sum of these values used within formula (11) amounts to $\nu(x)$. This value is then used to decide on min-optimality of the model within expression (10), where *max* is replaced by *min*.

Example 6
Example 3 defines an SMT-logic theory H that has three extended models

$$(\{b\},\nu_0),\ \ (\{b,|x \neq 0|\},\nu_1),\ \ (\{b,|x \neq 0|\},\nu_2),$$

where valuations ν_0,ν_1,ν_2 are chosen to assign x to 0, 1, 2, respectively. A sample OMT problem follows

$$\Big(H,\big\{\big(T_{\emptyset,\{x\},\{0,1,2\}},0;(x)\mapsto 1\big)\big\}\Big)$$

A solution for this OMT problem is an extended interpretation $(\{b\},\nu_0)$ – the one that minimizes the value of x.

5.2 Integer Linear Programming

An *integer linear program (IL-program)* (Papadimitriou and Steiglitz, 1982) was given in the introduction in (1). It is easy to see that the statement after the word `maximize` is an integer expression, whereas statements after the words `subject to` are integer constraints. We now provide an alternative definition to integer linear programs.

Definition 18
An IL-program is an ew-system of the form (H,S), where H is a theory in a nonnegative I-CSP-logic and S is an ew-condition $(T_{\emptyset,\Upsilon_H,\mathbb{Z}^+},0;\mathbf{c})$, where $\mathbf{c}$ is a coefficient function mapping variables in Υ_H to integers $\mathbb{Z}$. Extended optimal models to this system form *solutions to an IL-program.*

Intuitively, theory H corresponds to statements following `subject to` in (1), while the ew-condition S captures a statement following the word `maximize` in (1).

5.3 Optimizations in CAS Programs

We now turn our attention to constraint answer set programming. At first, we review optimization statements of answer set programming. We illustrate how they can be understood within CASP framework. For this task we utilize ew-systems: just as we utilized ew-systems to capture generalization from pw-MaxSAT to pw-MaxSMT realm. We then look into CLINGCON style optimization statements native to CASP framework. We illustrate that ew-systems are general enough to encapsulate CLINGCON programs with optimizations. Systems CLINGCON-2 (Ostrowski and Schaub, 2012) and CLINGCON-3 (Banbara et al., 2017) vary in the syntax of the languages they accept and algorithmic/implementation details. In case of this paper it is interesting to look at these systems in separation as they provide different support for optimization statements.

5.3.1 Optimizations in Logic Programming

We now review a definition of a logic program with weak constraints following the lines of Calimeri et al. (2013). A *weak constraint* has the form

$$:\sim a_1,\ldots,a_\ell,\ not\ a_{\ell+1},\ldots,\ not\ a_m[w@\ell], \tag{14}$$

where $m > 0$ and $a_1,\ldots,a_m$ are atoms, w (weight) is an integer, and l (level) is a positive integer. In the sequel, we abbreviate expression

$$:\sim a_1,\ldots,a_\ell,\ not\ a_{\ell+1},\ldots,\ not\ a_m \tag{15}$$

occurring in (14) as D and identify it with the propositional formula

$$a_1 \wedge \ldots \wedge a_\ell \wedge\ \neg a_{\ell+1} \wedge \ldots \wedge\ \neg a_m. \tag{16}$$

An *optimization program* (or *o-program*) over vocabulary σ is a pair (Π, W), where Π is a logic program over σ and W is a finite set of weak constraints over σ.

Let $\mathscr{P} = (\Pi, W)$ be an optimization program over vocabulary σ (intuitively, Π and W forms *hard* and *soft* fragments, respectively). By $\lambda(\mathscr{P})$ we denote the set of all levels associated with optimization program $\mathscr{P}$ constructed as $\{l \mid D[w@\ell] \in W\}$. Set X of atoms over σ is an *answer set* of $\mathscr{P}$ when it is an answer set of Π.

Definition 19 (*Optimal answer sets*)
Let X and X' be answer sets of $\mathscr{P}$. Answer set X' *dominates* X if there exists a level $l \in \lambda(\mathscr{P})$ such that following conditions are satisfied:

1. for any level l' that is greater than l the following equality holds

$$\sum_{D[w@l']\in W} w \cdot [X \models D] = \sum_{D[w@l']\in W} w \cdot [X' \models D]$$

2. the following inequality holds for level l

$$\sum_{D[w@\ell]\in W} w \cdot [X' \models D] < \sum_{D[w@\ell]\in W} w \cdot [X \models D]$$

An answer set X^* of $\mathscr{P}$ is *optimal* if there is no answer set X' of $\mathscr{P}$ that dominates X^*.

We now exemplify the definition of an optimization program. Let Π_1 be logic program (3). An optimal answer set of optimization program

$$(\Pi_1, \{:\sim a, not\ b. - 2@1\}) \tag{17}$$

is $\{a\}$.

It is worth noting that an alternative syntax is frequently used by answer set programming practitioners when they expresses optimization criteria:

$$\#minimize\{w_1@\ell_1 : lit_1,\ldots,w_n@\ell_n : lit_n\}, \tag{18}$$

where lit_i is either an atom a_i or an expression $not\ a_i$. This statement stands for n weak constraints

$$:\sim lit_1[w_1@\ell_1] \quad \ldots \quad :\sim lit_n[w_n@\ell_n].$$

Similarly, statement

$$\#maximize\{w_1@\ell_1 : lit_1,\ldots,w_n@\ell_n : lit_n\}, \tag{19}$$

stands for n weak constraints

$$:\sim lit_1[-w_1@\ell_1] \quad \ldots \quad :\sim lit_n[-w_n@\ell_n].$$

Lierler (2021) illustrated how o-programs can be identified with w-systems – a pre-cursor of ew-systems.

5.3.2 CLINGCON-2 *style optimizations*

Recall *minimize* statement (18). System CLINGCON-2 allows a user to write such statements using the following restrictions. All of these statements occurring in a program must either

- come with expressions lit_i ($1 \leq i \leq n$) constructed from regular atoms of CAS program, or
- come with expressions lit_i ($1 \leq i \leq n$) constructed from constraint variables stemming from irregular atoms of CAS program. Also, $w_i@\ell_i$ expressions are dropped in this case.[4]

Thus, a user might either impose optimization criteria that pertain regular atoms or constraint variables associated with irregular atoms but not both. At the same time we note that the CLINGCON-2 authors provide no declarative semantics for programs with optimizations, but rather explain the behavior of the systems by means of examples. In what follows we capture the semantics of two variants of CLINGCON-2 formally. We refer to programs of CLINGCON-2 supporting optimization statements over regular atoms CLINGCON-2.1 programs. We refer to programs of CLINGCON-2 supporting optimization statements over constraint variables CLINGCON-2.2 programs.

CLINGCON-2.1 *programs.* We now extend CAS programs with weak constraints in a similar way as MaxSMT extends SMT with "soft SMT clauses".

Definition 20 (*Optimization CAS program and their optimal (extended) answer sets*)
An *optimization CAS program* (or *oCAS-program*) over vocabulary σ, class $\mathscr{C}$ of constraints, and $(\sigma_c, \mathscr{C})$-denotation is an ew-system (P, W), where P is a CAS program over σ, $\mathscr{C}$ and $(\sigma_c, \mathscr{C})$-denotation, and W is a finite set of weak constraints over σ, where we identify weak constraints of the form $D[w@\ell]$ with ew-condition $(D, w@\ell)$ in RSMT-logic over σ, $\mathscr{C}$, and $(\sigma_c, \mathscr{C})$-denotation. We call oCAS-program CLINGCON-2 *style*, when there is an additional restriction on its weak constraints W to be over signature σ_r. Min-optimal models and min-optimal extended models of ew-system (P, W) form *optimal answer sets and optimal extended answer sets* of oCAS-program (P, W).

Note how the syntax of weak constraints supports optimizations that consider "propositional part" of a problem encoded within CASP. Indeed, coefficients functions of ew-conditions in oCAS-programs are assumed to be the zero functions. Thus, it is easy to see that any optimal extended answer set (X, ν) of some CAS program is such, whenever X is an optimal answer set of this program; the quality of evaluation ν of the extended answer set is immaterial. Recall how "minimize" statements (18) are abbreviations for the set of weak constraints (see Section 5.3.1). In this regard, CLINGCON-2.1 programs are captured by CLINGCON-2 style oCAS-programs, in other words, the definitions of optimal answer sets and optimal extended answer sets of oCAS-program capture formally the semantics of CLINGCON-2.1 programs.

[4] In case, when this restriction is not satisfied the system CLINGCON-2 outputs the following message *ERROR: Can not optimize asp and csp at the same time!*

We also note that this observation allows us to utilize pw-MaxSMT solvers (such as, for example, solver νZ discussed here) for finding solutions to CLINGCON-2.1 programs. Indeed, the essential difference between CLINGCON-2.1 programs and pw-MaxSMT formalism boils down to the first component of the pairs capturing these objects. In one case we deal with CAS programs, in another with SMT formulas. Theory behind SMT-based CASP solver EZSMT (Lierler and Susman, 2017; Shen and Lierler, 2018) relies on the established link between these two. Here, we paved the way at enhancing EZSMT with the support for optimization statements.

CLINGCON-2.2 *programs.* We now elaborate on CLINGCON-2.2 programs. We start by quoting Ostrowski and Schaub (2012), who provide examples of optimization statements with constraint variables and informal discussions in order to illustrate their behavior. We supplement this quote with additional comments using square brackets to ease the understanding of the narrative. The CLINGCON-2.2 programs allow expressions of the kind

```
$maximize{work(A) : person(A)}.
```

and its developers say that this is an instance of

> a maximize statement over constraint variables. This is also a new feature of clingcon. We maximize the sum over a set of variables and/or expressions. In this case, we try to maximize
>
> ```
> work(adam) $+work(smith) $+ work(lea) $+ work(john).
> ```
>
> [During grounding (see, for instance, an account for grounder GRINGO (Gebser et al., 2007b)) – a process that is a common first step in solving ASP programs, where ASP variables are instantiated for suitable object constants – it was established that ASP variable *A* may take four values, namely, *adam, smith, lea* and *john*.] ... To find a constraint optimal solution, we have to combine the enumeration techniques of CLASP [by Gebser et al. (2007a) – answer set solver within CLINGCON] with the ones from the CP solver. Therefore, when we first encounter a full propositional assignment, we search for an optimal (w.r.t. to the optimize statement) assignment of the constraint variables using the search engine of the CP solver. Let us explain this with the following constraint logic program.
>
> ```
> $domain(1..100).
> a :- x $* x $< 25.
> $minimize{x}.
> ```
>
> Assume clasp has computed the full assignment $\{Fx\$*x\$ < 25, Fa\}$ [irregular atom named $x\$*x\$ < 25$ and regular atom a are assigned value *false*; intuitively the mentioned irregular atom is mapped into inequality $x*x < 25$ with constraint variable x]. Afterwards, we search for the constraint optimal solution to the constraint variable x, which yields $\{x \to 5\}$. Given this optimal assignment, a constraint can be added to the CP solver that all further solutions shall be below/above this optimum $(x < 5)$. This constraint will now restrict all further solutions to be "better". We enumerate further solutions, using the enumeration techniques of CLASP. So the next assignment is $\{Tx\$*x\$ < 25, Ta\}$ and the CP solver finds the optimal constraint variable assignment $\{x \to 1\}$. Each new solution restricts the set of further solutions, so our constraint is changed to $(x\$ < 1)$, which then allows no further solutions to be found.

To formalize the claims of this quote let us capture CLINGCON-2.2 programs as a CAS program P extended with an expression of the form

$$\$minimize\{lit_1, \ldots, lit_n\}, \tag{20}$$

where lit_i $(1 \leq i \leq n)$ is a constraint variable stemming from irregular atoms of the considered CAS program. Ew-systems can be used to characterize CLINGCON-2.2 programs as follows.

Definition 21 (CLINGCON-2.2 *programs and their optimal answer sets*)
A CLINGCON-2.2 program P (where P is a CAS program over σ, $\mathscr{C}$ and $(\sigma_c, \mathscr{C})$-denotation) extended with minimization statement (20) is ew-system (P, W), where W is a single ew-condition $(T_{\sigma_P, \Upsilon_P, \Delta_P}, 0; c)$ so that c is a coefficients function that to every constraint variable in Υ_P assigns 1 when it appears in (20) and 0 otherwise. Models of this ew-systems are called *answer sets* of P, while min-optimal models are called *optimal answer sets* of P.

Note how CLINGCON-2.1 and CLINGON-2.2 programs allow the user to either optimize propositional side of a problem or constraint side of the problem but never both. We now provide a definition for CAS programs with optimizations restoring to the method adopted earlier in stating the definition for gpw-MaxSMT problem. This allows us to incorporate quality of "constraint" part of the solution into assessment of overall quality of considered solution. This way quality of propositional and constraint part of solution is taken into account.

Definition 22 (*CAS program with generalized optimizations; their optimal (extended) answer sets*)

A *CAS program with generalized optimizations* over vocabulary σ, class $\mathscr{C}$ of constraints, and $(\sigma_c, \mathscr{C})$-denotation is defined as ew-system (H, S), where H is a theory in CAS-logic over σ, $\mathscr{C}$, and $(\sigma_c, \mathscr{C})$-denotation and S is a collection of ew-conditions, whose theories are in RSMT-logic over σ, $\mathscr{C}$, and $(\sigma_c, \mathscr{C})$-denotation. Min-optimal models and min-optimal extended models of ew-system (H, S) form *optimal answer sets and optimal extended answer sets* of CAS program with generalized optimizations (H, S).

Note how in this definition ew-conditions are more general than in oCAS-programs or CLINGCON 2.2 programs.

5.3.3 CLINGCON-*3 style optimizations*

Banbara et al. (2017) introduce optimizations supported within CLINGCON-3. They propose minimize statements for CAS programs that have the form

$$\$minimize\{b_1 \cdot x_1 + c_1 @ \ell_1,\ \ldots,\ b_n \cdot x_n + c_n @ \ell_n\}, \tag{21}$$

where b_i and c_i are integers, x_i are variables stemming from the constraint part of the program, and ℓ_i is a level – positive integer. In addition, for any two expressions $b \cdot x + c @ \ell$ and $b' \cdot x' + c' @ \ell$ occurring in (21) variables x and x' are distinct. Such minimize statements induce optimal extended answer sets as follows. For the remainder of this subsection, let P be a CAS program P' extended with minimize statement of the form (21). Any (extended) answer set of P' is an *(extended) answer set* of P. For a variable assignment ν and an integer l,

$$\sum_{l}^{\nu} = \sum_{b \cdot x + c @ \ell \in (21)} b \cdot \nu(x) + c.$$

Definition 23 (*Optimal answer sets of CAS programs due to Banbara et al. (2017)*)
Let (X, ν) and (X', ν') be extended answer sets of P. Extended answer set (X', ν') *dominates* (X, ν) if there exists a level $l \in \{\ell_1, \ldots, \ell_n\}$ ($\ell_1, \ldots, \ell_n$ are levels occurring in (21)) such that

1. for any level $l' \in \{\ell_1, \ldots, \ell_n\}$ that is greater than l the following equality holds

$$\sum_{l'}^{\nu} = \sum_{l'}^{\nu'},$$

2. the following inequality holds for level l

$$\sum_{l}^{\nu'} < \sum_{l}^{\nu},$$

Extended answer set (X^*, ν^*) of P is optimal if there is no extended answer set (X', ν') that dominates (X^*, ν^*).

For every level l appearing in (21), by

- w_l we denote the sum $\sum_{b\cdot x + c@\ell \in \text{ (21)}} c$ of constant terms in integer expressions associated with each level;
- c_l we denote coefficient function from $\Upsilon_{P'}$ to $\mathbb{Z}$ that maps each variable x occurring in $\Upsilon_{P'}$ and in expression $b \cdot x + c@\ell$ in (21) to b, while all other variables in $\Upsilon_{P'}$ are mapped to 0.

As earlier we can identify any CAS program with the CAS-logic module. For the CAS program P' extended with the minimize statement (21), we identify (21) with the set consisting of the following ew-conditions

- for every level l appearing in (21), ew-condition $(T_{\sigma_{P'}, \Upsilon_{P'}, \Delta_{P'}}, w_l@\ell)$; and
- for every level l appearing in (21), ew-condition $(T_{\sigma_{P'}, \Upsilon_{P'}, \Delta_{P'}}, 0; c_l@\ell)$.

Once more we can use ew-systems to provide an alternative definition for CAS programs with minimization statements of the kind introduced in this subsection.

Proposition 3
Let P be a CAS program P' extended with minimize statements of the form (21) over vocabulary σ, class $\mathscr{C}$ of constraints, and $(\sigma_c, \mathscr{C})$-denotation. Min-optimal extended models of ew-system (P', S) — where S is a collection of ew-conditions identified/associated with (21) of P — are optimal answer sets of P.

6 Formal Properties of Ew-systems

Lierler (2021) stated many interesting formal results for the case of w-systems; Lierler (2022) presented proofs for these results. Many of these results/proofs can be lifted to the case of ew-systems. Here we present a series of formal results about ew-systems. Word *Property* denotes the results that follow rather immediately from the definitions of a model/optimal (extended) model.

Property 1
Any two ew-systems with the same hard theory have the same models/extended model.

Due to this property when stating the results for ew-systems that share the same hard theory, we only focus on optimal and min-optimal (extended) models.

Property 2
Any model/extended model of ew-system of the form $(\mathscr{H}, \emptyset)$ is optimal/min-optimal.

Property 3
Optimal/min-optimal models of the following ew-systems coincide

- ew-system $\mathscr{W}$ and
- ew-system resulting from $\mathscr{W}$ by dropping all of its w-conditions whose weight is 0.

Property 4
Optimal/min-optimal models of the following ew-systems coincide

- ew-system $\mathscr{W}$ and
- ew-system resulting from $\mathscr{W}$ by replacing each of its ew-conditions of the form $(T, w, \mathbf{c}@\ell)$ with ew-condition $(T, w@\ell)$.

This property points at the fact that **c** component of ew-conditions are only relevant when optimality of *extended* models is considered.

We now state simple properties that pertain extended models of ew-systems.

Property 5
Let $\mathscr{W}$ be an ew-system, whose ew-conditions have special form $(T, w@\ell)$. An extended model (I, ν) of $\mathscr{W}$ is optimal/min-optimal if and only if I is an optimal/min-optimal model of $\mathscr{W}$.

Property 6
Optimal/min-optimal extended models of the following ew-systems coincide

- ew-system $\mathscr{W}$ and
- ew-system resulting from $\mathscr{W}$ by dropping all of its ew-conditions whose form is $(T, 0@\ell)$

We call an ew-system $\mathscr{W}$ *level-normal*, when we can construct the sequence of numbers $1, 2, \ldots, |\lambda(\mathscr{W})|$ from the elements in $\lambda(\mathscr{W})$. Lierler (2021) stated propositions in spirit of Propositions 4 and 5 presented below for the case of w-systems. Here we lift these results to the case of ew-systems.

Proposition 4
Optimal/min-optimal models/extended models of the following ew-systems coincide

- ew-system $\mathscr{W}$ and
- the level-normal ew-system constructed from $\mathscr{W}$ by replacing each level ℓ_i occurring in its ew-conditions with its ascending sequence order number i, where we arrange elements in $\lambda(\mathscr{W})$ in a sequence in ascending order $\ell_1, \ell_2, \ldots \ell_{|\lambda(\mathscr{W})|}$.

Proposition 5
For an ew-system $\mathscr{W} = (\mathscr{H}, \mathscr{L})$, if every level $l \in \lambda(\mathscr{W})$ is such that for any distinct models I and I' of $\mathscr{W}$ the equality

$$\sum_{B \in \mathscr{W}_l} [I \models B] = \sum_{B \in \mathscr{W}_l} [I' \models B] \tag{22}$$

holds then optimal/min-optimal models of ew-systems $\mathscr{W}$ and $(\mathscr{H}, \emptyset)$ coincide. Or, in other words, any model of $\mathscr{W}$ is also optimal and min-optimal model.

The proposition above formulated for the case of extended models follows.

Proposition 6
For an ew-system $\mathscr{W} = (\mathscr{H}, \mathscr{Z})$, if every level $l \in \lambda(\mathscr{W})$ is such that for any distinct extended models (I, ν) and (I', ν') of $\mathscr{W}$ the equality

$$\sum_{B \in \mathscr{W}_l} ([I \models B] + [(I, \nu) \models B]) = \sum_{B \in \mathscr{W}_l} ([I' \models B] + [(I', \nu') \models B]) \tag{23}$$

holds then optimal/min-optimal extended models of ew-systems $\mathscr{W}$ and $(\mathscr{H}, \emptyset)$ coincide. Or, in other words, any extended model of $\mathscr{W}$ is also optimal and min-optimal model.

Let $\mathscr{W} = (\mathscr{H}, \mathscr{Z})$ be an ew-system. For a set S of ew-conditions, by $\mathscr{W}[\backslash S]$ we denote the ew-system $(\mathscr{H}, \mathscr{Z} \setminus S)$.

Proposition 7
For a ew-system $\mathscr{W} = (\mathscr{H}, \mathscr{Z})$, if there is a set $S \subseteq \mathscr{Z}$ of ew-conditions all sharing the same level l such that for any distinct $l^{\uparrow}$-optimal/min-optimal models I and I' of $\mathscr{W}$ (or any distinct models I and I' of $\mathscr{W}$ in case $l^{\uparrow}$ is undefined) the equality (22), where $\mathscr{W}_l$ is replaced by S, holds then $\mathscr{W}$ has the same optimal/min-optimal models as $\mathscr{W}[\backslash S]$.

We can formulate a similar claim for the case of extended models.

Proposition 8
For an ew-system $\mathscr{W} = (\mathscr{H}, \mathscr{Z})$, if there is a set $S \subseteq \mathscr{Z}$ of ew-conditions all sharing the same level such that for any distinct $l^{\uparrow}$-optimal/min-optimal extended models (I, ν) and (I', ν') of $\mathscr{W}$ (or any distinct extended models (I, ν) and (I', ν') of $\mathscr{W}$ in case $l^{\uparrow}$ is undefined) the equality (23), where $\mathscr{W}_l$ is replaced by S, holds then $\mathscr{W}$ has the same optimal/min-optimal extended models as $\mathscr{W}[\backslash S]$.

For a coefficients function $\mathbf{c}$ mapping variables into reals by $\mathbf{c}^{-1}$ we denote a function on the same set of variables as $\mathbf{c}$ defined as follows

$$\mathbf{c}^{-1}(x) = -1 \cdot \mathbf{c}(x).$$

For ew-condition $(T, w; \mathbf{c}@\ell)$, we define two mappings into related ew-conditions

$$(T, w; \mathbf{c}@\ell)^{-1\cdot} = (T, -1 \cdot w; \mathbf{c}@\ell),$$
$$(T, w; \mathbf{c}@\ell)^{-1\cdot -1\cdot} = (T, -1 \cdot w; \mathbf{c}^{-1}@\ell).$$

For ew-system $(\mathscr{H}, \{B_1, \ldots, B_n\})$, we define two mappings into related ew-systems using concepts above

$$(\mathscr{H}, \{B_1, \ldots, B_n\})^{-1\cdot} = (\mathscr{H}, \{{B_1}^{-1\cdot}, \ldots, {B_n}^{-1\cdot}\})$$
$$(\mathscr{H}, \{B_1, \ldots, B_n\})^{-1\cdot -1\cdot} = (\mathscr{H}, \{{B_1}^{-1\cdot -1\cdot}, \ldots, {B_n}^{-1\cdot -1\cdot}\})$$

With this newly introduced notation we can now claim the relation between optimal and min-optimal (extended) models of ew-systems.

Proposition 9
For an ew-system $\mathscr{W}$, the optimal models (min-optimal models) of $\mathscr{W}$ coincide with the min-optimal models (optimal models) of $\mathscr{W}^{-1\cdot}$.

Proposition 10
For an ew-system $\mathscr{W}$, the extended optimal models (extended min-optimal models) of $\mathscr{W}$ coincide with the extended min-optimal models (extended optimal models) of $\mathscr{W}^{-1\cdot -1\cdot}$.

Eliminating Negative (or Positive) Weights We call e-logics $\mathscr{L}$ and $\mathscr{L}'$ *compatible* when their vocabularies, domains, and variables coincide, i.e., $\sigma_{\mathscr{L}} = \sigma_{\mathscr{L}'}$, $\Delta_{\mathscr{L}} = \Delta_{\mathscr{L}'}$, and $\Upsilon_{\mathscr{L}} = \Upsilon_{\mathscr{L}'}$. Let $\mathscr{L}$ and $\mathscr{L}'$ be compatible logics, and T and T' be theories in these logics, respectively. We call a theory T (and a w-condition $(T, w; \mathbf{c}@\ell)$) *equivalent to* a theory T' (and a w-condition $(T', w; \mathbf{c}@\ell)$, respectively), when $sem(T) = sem(T')$.

Property 7
Models and optimal/min-optimal models/extended models of ew-systems

$$(\{T_1, \ldots, T_n\}, \{B_1, \ldots, B_m\}) \text{ and } (\{T'_1, \ldots, T'_n\}, \{B'_1, \ldots, B'_m\})$$

coincide when (i) T_i and T'_i $(1 \leq i \leq n)$ are equivalent theories, and (ii) B_i and B'_i $(1 \leq i \leq m)$ are equivalent ew-conditions.

For a theory T of e-logic $\mathscr{L}$, we call a theory $\overline{T}$ in e-logic $\mathscr{L}'$, compatible to $\mathscr{L}$, *complementary* when (i) $sem(T) \cap sem(\overline{T}) = \emptyset$, and (ii) $sem(T) \cup sem(\overline{T}) = Int(\sigma_{\mathscr{L}}, \Upsilon_{\mathscr{L}}, \Delta_{\mathscr{L}})$.

Let $(T, w; \mathbf{c}@\ell)$ be an ew-condition; consider the following definitions:

$$(T, w; \mathbf{c}@\ell)^{+} = \begin{cases} (T, w; \mathbf{c}@\ell) & \text{when } w \geq 0, \text{ otherwise} \\ (\overline{T}, -1 \cdot w; \mathbf{c}@\ell) \end{cases}$$

$$(T, w; \mathbf{c}@\ell)^{-} = \begin{cases} (T, w; \mathbf{c}@\ell) & \text{when } w \leq 0, \text{ otherwise} \\ (\overline{T}, -1 \cdot w; \mathbf{c}@\ell) \end{cases}$$

$$(T, w; \mathbf{c}@\ell)^{+;+} = \begin{cases} (T, w; \mathbf{c}@\ell) & \text{when } w \geq 0, \text{ otherwise} \\ (\overline{T}, -1 \cdot w; \mathbf{c}^{-1}@\ell) \end{cases}$$

$$(T, w; \mathbf{c}@\ell)^{-;-} = \begin{cases} (T, w; \mathbf{c}@\ell) & \text{when } w \leq 0, \text{ otherwise} \\ (\overline{T}, -1 \cdot w; \mathbf{c}^{-1}@\ell) \end{cases}$$

where $\overline{T}$ denotes some theory complement to T.

For an ew-system $(\mathscr{H}, \{B_1, \ldots, B_m\})$, we define

$$\begin{array}{l} (\mathscr{H}, \{B_1, \ldots, B_n\})^{+} = (\mathscr{H}, \{{B_1}^{+}, \ldots, {B_n}^{+}\}), \\ (\mathscr{H}, \{B_1, \ldots, B_n\})^{-} = (\mathscr{H}, \{{B_1}^{-}, \ldots, {B_n}^{-}\}), \\ (\mathscr{H}, \{B_1, \ldots, B_n\})^{+;+} = (\mathscr{H}, \{{B_1}^{+;+}, \ldots, {B_n}^{+;+}\}), \\ (\mathscr{H}, \{B_1, \ldots, B_n\})^{-;-} = (\mathscr{H}, \{{B_1}^{-;-}, \ldots, {B_n}^{-;-}\}). \end{array} \tag{24}$$

Proposition 11
Optimal/min-optimal models of ew-systems $\mathscr{W}$, $\mathscr{W}^{+}$, $\mathscr{W}^{-}$, $\mathscr{W}^{+;+}$, $\mathscr{W}^{-;-}$ coincide.

This proposition can be seen as an immediate consequence of the following result and Property 4:

Proposition 12
Optimal/min-optimal models of ew-systems

- $(\mathscr{H}, \{(T, w@\ell)\} \cup \mathscr{Z})$ and
- $(\mathscr{H}, \{(\overline{T}, -1 \cdot w@\ell)\} \cup \mathscr{Z})$

coincide.

Proposition 13
Optimal/min-optimal extended models of ew-systems $\mathscr{W}$, $\mathscr{W}^{+;+}$, $\mathscr{W}^{-;-}$ coincide.

This proposition can be seen as an immediate consequence of the following result:

Proposition 14
Optimal/min-optimal extended models of ew-systems

- $(\mathscr{H}, \{(T, w; \mathbf{c}@\ell)\} \cup \mathscr{Z})$ and
- $(\mathscr{H}, \{(\overline{T}, -1 \cdot w; \mathbf{c}^{-1}@\ell)\} \cup \mathscr{Z})$

coincide.

The notation and results of the section on *Eliminating Levels* by Lierler (2021; 2022) can be lifted to the case of ew-systems and their optimal/min-optimal models in a straightforward manner (similar as the results for eliminating negative/positive weights are lifted here). Hence, we omit the review of these results. Yet, the "cost" expression associated with the second member of extended model is more complex so that the role of the levels seem to go beyond syntactic sugar when extended models are considered.

7 Proofs

Many of the formal properties of ew-systems presented in Section 6 echo similar results for w-systems — a precursor of ew-systems introduced by Lierler (2021). Lierler (2022) presented proofs for these results for w-systems. The logic and structure of proofs for the case of ew-systems follows the proofs for the case of w-systems. In this section, we often remark on the connection to the proofs by Lierler (2022) and point at any details worth noting.

Just as in case for w-systems, we focus on results for optimal (extended) models only, as the arguments for min-optimal (extended) models follow the same lines. Given recursive Definitions 10 and 11 of l-(min)-optimal (extended) models, inductive argument is a common technique in proof construction about properties of such models. In particular, the *induction on levels of a considered ew-system* $\mathscr{W}$, where we assume elements in $\lambda(\mathscr{W})$ to be arranged in the descending order $m_1, \ldots m_n$ ($n = |\lambda(\mathscr{W})|$); so that the base case is illustrated for the greatest level m_1, whereas inductive hypothesis is assumed for level m_i and then illustrated to hold for level m_{i+1}. Note how, ${m_{i+1}}^{\uparrow} = m_i$.

Proof of Proposition 1
The statement of Proposition 1 echos the statement of Proposition 1 by Lierler (2022) for the case of w-systems. The proofs of the two claims (i) Definitions 10 and 12 are equivalent; and (ii) Definitions 11 and 13 are equivalent follow the lines of the proof provided for Proposition 1 by Lierler (2022). In fact, for the claim (i) we can repeat the proof practically verbatim modulo the understanding that in place of w-system $\mathscr{W}$ we consider ew-system $\mathscr{W}$ as well as in place of w-conditions of $\mathscr{W}$ we consider ew-conditions of $\mathscr{W}$. For the claim (ii), in addition to the proof of Proposition 1 by Lierler (2022) will have to refer to extended interpretations in place of interpretations and to summations of the form

$$\sum_{B \in \mathscr{W}_l} ([I \models B] + [(I, \nu) \models B])$$

in place of summations of the form

$$\sum_{B \in \mathscr{W}_l} ([I \models B].$$

Claims of Lemmas 1, 2, and 3 by Lierler (2022) hold for the case of ew-systems and their l-optimal models and l-optimal extended models. □

Proof of Proposition 2
Let S be a weighted MaxSMT problem and $\mathscr{W}_S = (T_{\sigma_S, \Upsilon_S, \Delta_S}, S)$ be a respective ew-system — where each element in S is understood as an ew-condition, whose theory is in SMT-logic. Consider an arbitrary interpretation $I^* \in Int(\sigma_S)$. We show that I^* is a solution to weighted MaxSMT problem S if and only if I^* is an optimal models of $\mathscr{W}_S$. Per Definition 14, interpretation I^* is a solution to S if and only if

$$I^* = \arg\max_I \sum_{(\mathscr{F}, w) \in S} w \cdot [I \models \mathscr{F}],$$

where I ranges over all interpretations in $Int(\sigma_S)$. By the definition of $T_{\sigma_S, \Upsilon_S, \Delta_S}$, any interpretation in $Int(\sigma_S)$ is its model. Per Definition 10, interpretation I^* is an optimal model of $\mathscr{W}_S$ if and only if

$$I^* = arg\max_I \sum_{(\mathscr{F}, w) \in S} [I \models (\mathscr{F}, w)],$$

where I ranges over all interpretations in $Int(\sigma_S)$.

Taking definitions (7) and (12) into account and an understanding that each element in S can be seen as an ew-condition, whose theory is in SMT-logic, we conclude that for any interpretation I in $Int(\sigma_S)$

$$\sum_{(\mathscr{F}, w) \in S} w \cdot [I \models \mathscr{F}] = \sum_{(\mathscr{F}, w) \in S} [I \models (\mathscr{F}, w)].$$

□

Proof of Proposition 3
Let P be a CAS program P' extended with minimize statements of the form (21) over vocabulary σ, class $\mathscr{C}$ of constraints, and $(\sigma_c, \mathscr{C})$-denotation. Consider an extended answer set (X^*, v^*) of P. We show that (X^*, v^*) is optimal extended answer set of P if and only if (X^*, v^*) is min-optimal extended model of ew-system (P', S) — where S is a collection of ew-conditions identified/associated with (21) of P.

It is easy to see that any extended answer set of P is an extended model of (P', S). Thus, the proof focuses on optimality condition.

Per Definition 19, (X^*, v^*) is an optimal extended answer set if and only if there is no extended answer set (X', v') that dominates (X^*, v^*). In other words, any answer set (X', v') is such that for every level l occurring in (21) either

1. there exists a level l' occurring in (21) that is greater than l and the following inequality holds

$$v^*_{l'} \neq v'_{l'},$$

or

2. the following inequality holds for level l

$$\sum_{l}^{\nu'} \geq \sum_{l}^{\nu^*},$$

Per Definition 13, (X^*, ν^*) is min-optimal extended model of ew-system (P', S) if and only if there is no extended model (I', ν') of (P', S) that min-dominates (X^*, ν^*). In other words, any extended model (X', ν') is such that for every level $l \in \lambda((P', S))$ either

1. there exists a level $l' \in \lambda((P', S))$ that is greater than l and the following inequality holds

$$\sum_{B \in (P', S)_{l'}} ([X^* \models B] + [(X^*, \nu^*) \models B]) \neq \sum_{B \in (P', S)_{l'}} ([X' \models B] + [(X', \nu') \models B])$$

or

2. the following inequality holds for level l

$$\sum_{B \in (P', S)_{l}} ([X' \models B] + [(X', \nu') \models B]) \geq \sum_{B \in (P', S)_{l}} ([X^* \models B] + [(X^*, \nu^*) \models B])$$

We first observe that by the construction of S any level l occurs in (21) if and only if $l \in \lambda((P', S))$. Recall that any extended answer set of P is an extended model of (P', S). It is now sufficient to show that given any extended model (X, ν) of (P', S), the following equality holds for arbitrary level $l \in \lambda((P', S))$:

$$\sum_{l}^{\nu} = \sum_{B \in (P', S)_{l}} ([X \models B] + [(X, \nu) \models B]). \tag{25}$$

Indeed, per definition of $\sum_l^\nu$

$$\sum_{l}^{\nu} = \sum_{b \cdot x + c @ \ell \text{ in } (21)} (b \cdot \nu(x) + c) = \sum_{b \cdot x + c @ \ell \text{ in } (21)} b \cdot \nu(x) + \sum_{b \cdot x + c @ \ell \text{ in } (21)} c. \tag{26}$$

Per S construction, $(P', S)_l$ consists of two ew-conditions

- $(T_{\sigma_{P'}, \Upsilon_{P'}, \Delta_{P'}}, w_l @ \ell)$; and
- $(T_{\sigma_{P'}, \Upsilon_{P'}, \Delta_{P'}}, 0; c_l @ \ell)$.

It is easy to see that

$$[(X, \nu) \models (T_{\sigma_{P'}, \Upsilon_{P'}, \Delta_{P'}}, w_l @ \ell)] = 0,$$
$$[X \models (T_{\sigma_{P'}, \Upsilon_{P'}, \Delta_{P'}}, 0; c_l @ \ell)] = 0.$$

Per definitions of w_l and c_l, we derive that

$$[X \models (T_{\sigma_{P'}, \Upsilon_{P'}, \Delta_{P'}}, w_l @ \ell)] = w_l = \sum_{b \cdot x + c @ \ell \text{ in } (21)} c,$$

and

$$[(X, \nu) \models (T_{\sigma_{P'}, \Upsilon_{P'}, \Delta_{P'}}, 0; c_l @ \ell)] = \sum_{x \in \Upsilon_{P'}} \nu(x) \cdot c_l(x) = \sum_{b \cdot x + c @ \ell \text{ in } (21)} b \cdot \nu(x).$$

Consequently,

$$\sum_{B \in (P', S)_{l}} ([X \models B] + [(X, \nu) \models B]) = \sum_{b \cdot x + c @ \ell \text{ in } (21)} b \cdot \nu(x) + \sum_{b \cdot x + c @ \ell \text{ in } (21)} c. \tag{27}$$

Equality (25) follows immediately from (26) and (27). □

Proof of Proposition 4 follows from the fact that the numeric value of any level itself is inessential in the key computations associated with establishing optimal (extended) models. Rather, the order of levels with respect to greater relation matters (recall the definition of $(\cdot)^{\uparrow}$ operation). It is easy to see that changing levels of the w-conditions using the procedure described in this proposition preserves original order of the levels with respect to greater relation.

Propositions 5 and 6 follow immediately from Propositions 7 and 8, respectively. The statement of Proposition 7 echos the statement of Proposition 8 by Lierler (2022) for the case of w-systems. The statement of Proposition 8 lifts the statement of Proposition 7 from the case of models to the case of extended models. The proofs of Propositions 7 and 8 follow the lines of the proof provided for Proposition 8 by Lierler (2022) modulo similar provisions as pointed at in the beginning of this section in *Proof of Proposition 1 (sketch)*.

The statement and proof of Proposition 9 echos the statement and proof of Proposition 9 by Lierler (2022) for the case of w-systems. The structure of the following proof is in spirit of the proof of Proposition 9 by Lierler (2022) and, yet, we state some of its details here as mapping $(\cdot)^{-1\cdot-1\cdot}$ is unique to this work.

Proof of Proposition 10
To show that the extended optimal models of $\mathscr{W}$ coincide with the min-optimal extended models of $\mathscr{W}^{-1\cdot-1\cdot}$, it is sufficient to show that for any level in $\lambda(\mathscr{W})$, l-optimal extended models of $\mathscr{W}$ coincide with l-min-optimal extended models of $\mathscr{W}^{-1\cdot-1\cdot}$. We first note that extended models of $\mathscr{W}$ and $\mathscr{W}^{-1\cdot-1\cdot}$ coincide. By induction on levels of $\mathscr{W}$.

Base case. l is the greatest level. An extended model (I^*, ν^*) of $\mathscr{W}$ is *l-optimal* if and only if (I^*, ν^*) satisfies equation (10), where (I, ν) ranges over extended models of $\mathscr{W}$. It is easy to see that we can rewrite this equation as

$$(I^*, \nu^*) = arg\min_{(I,\nu)} \sum_{B \in \mathscr{W}_l} \Big(-1 \cdot [I \models B] + (-1 \cdot [(I, \nu) \models B]) \Big).$$

It immediately follows from the construction of $\mathscr{W}^{-1\cdot-1\cdot}$ that this equation can be rewritten as

$$(I^*, \nu^*) = arg\min_{(I,\nu)} \sum_{B \in \mathscr{W}_l^{-1\cdot-1\cdot}} ([I \models B] + [(I, \nu) \models B]).$$

Thus (I^*, ν^*) is an l-min-optimal extended model of $\mathscr{W}^{-1\cdot-1\cdot}$ as the equation above is exactly the one from the definition of l-min-optimal extended models of $\mathscr{W}^{-1\cdot-1\cdot}$; plus recall that extended models of $\mathscr{W}$ and $\mathscr{W}^{-1\cdot-1\cdot}$ coincide.

Inductive case argument follows similar lines. □

Propositions 11 and 13 follow from Propositions 12 and 14, respectively. Proofs of Propositions 12 and 14 follow the lines of proof of Proposition 11 stated by Lierler (2022).

8 Conclusions and Acknowledgments

We trust that the proposed unifying framework of ew-systems will allow developers of distinct automated paradigms to better grasp similarities and differences of the kind of optimization criteria their paradigms support. In practice, translational approaches are popular in devising solvers. These approaches rely on the established relationships between automated reasoning paradigms. In particular, rather than devising a unique search algorithm for a paradigm of interest, researchers propose a translation from this "source" paradigm to another "target" framework.

As a result solvers for the target framework are used to find solutions for a problem encoded in the source paradigm. This work is a stepping stone towards extending these translational approaches with the support for optimization statements. We proposed the extension of abstract modular systems to extended weighted systems in a way that modern approaches to optimizations stemming from a variety of different logic based formalisms can be studied in unified terminological ways so that their differences and similarities become clear not only on intuitive but also formal level. These ew-systems allowed us to provide generalizations for the family of MaxSMT problems that incorporate optimizations over theory/constraint elements of these problems in addition to their propositional side. The framework also provides immediate support for the concept of levels of optimization criteria. We also provided formal semantics for two variants of CLINGCON-2 programs that mimic the behavior of their informal descriptions in the literature. We trust that establishing clear link between optimization statements, criteria, and solving in distinct automated reasoning subfields is a truly fruitful endeavor allowing a streamlined cross-fertilization between the fields. The EZSMT (Shen and Lierler, 2018) system is a translational constraint answer set solver that translates its programs into satisfiability modulo theories formulas. We trust that results obtained here lay the groundwork for extending a "translational" solver EZSMT with the support for optimization statements.

The work was partially supported by NSF grant 1707371. We are grateful to anonymous reviewers for valuable comments on this paper.

Competing interests: The author(s) declare none.

References

ALVIANO, M. 2018. Algorithms for solving optimization problems in answer set programming. *Intelligenza Artificiale*, *12*, 1–14.

ALVIANO, M., ROMERO, J., AND SCHAUB, T. Preference relations by approximation. In *KR* 2018, pp. 2–11. AAAI Press.

BANBARA, M., KAUFMANN, B., OSTROWSKI, M., AND SCHAUB, T. 2017. Clingcon: The next generation. *Theory and Practice of Logic Programming*, *17*, 4, 408–461.

BARRETT, C., FONTAINE, P., AND TINELLI, C. 2016. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`.

BARRETT, C., STUMP, A., AND TINELLI, C. The SMT-LIB Standard: Version 2.0. In GUPTA, A. AND KROENING, D., editors, *Proceedings of the 8th International Workshop on Satisfiability Modulo Theories (Edinburgh, UK)* 2010.

BARRETT, C. AND TINELLI, C. Satisfiability modulo theories. In CLARKE, E., HENZINGER, T., AND VEITH, H., editors, *Handbook of Model Checking* 2014. Springer.

BJØRNER, N., PHAN, A.-D., AND FLECKENSTEIN, L. νz - an optimizing smt solver. In BAIER, C. AND TINELLI, C., editors, *Tools and Algorithms for the Construction and Analysis of Systems* 2015, pp. 194–199, Berlin, Heidelberg. Springer Berlin Heidelberg.

BREWKA, G., DELGRANDE, J. P., ROMERO, J., AND SCHAUB, T. asprin: Customizing answer set preferences without a headache. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. 2015, pp. 1467–1474.

BREWKA, G. AND EITER, T. Equilibria in heterogeneous nonmonotonic multi-context systems. In *Proceedings of National Conference on Artificial Intelligence, AAAI 2007* 2007, pp. 385–390.

CALIMERI, F., FABER, W., GEBSER, M., IANNI, G., KAMINSKI, R., KRENNWALLNER, T., LEONE, N., RICCA, F., AND SCHAUB, T. 2013. Asp-core-2 input language format. *URL https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.03c.pdf,*.

DE MOURA, L. AND BJØRNER, N. Z3: An efficient smt solver. In RAMAKRISHNAN, C. R. AND REHOF,

J., editors, *Tools and Algorithms for the Construction and Analysis of Systems* 2008, pp. 337–340, Berlin, Heidelberg. Springer Berlin Heidelberg.

GEBSER, M., KAMINSKI, R., KAUFMANN, B., OSTROWSKI, M., SCHAUB, T., AND WANKO, P. Theory Solving Made Easy with Clingo 5. In CARRO, M., KING, A., SAEEDLOEI, N., AND VOS, M. D., editors, *Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016)* 2016, volume 52 of *OpenAccess Series in Informatics (OASIcs)*, pp. 2:1–2:15, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

GEBSER, M., KAUFMANN, B., NEUMANN, A., AND SCHAUB, T. Conflict-driven answer set solving. In *Proceedings of 20th International Joint Conference on Artificial Intelligence (IJCAI'07)* 2007a, pp. 386–392. MIT Press.

GEBSER, M., SCHAUB, T., AND THIELE, S. Gringo: A new grounder for answer set programming. In *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning* 2007b, pp. 266–271.

LIERLER, Y. 2014. Relating constraint answer set programming languages and algorithms. *Artificial Intelligence*, *207C*, 1–22.

LIERLER, Y. An abstract view on optimizations in SAT and ASP. In *Proceedings of the 17th European Conference on Logics in Artificial Intelligence (JELIA)* 2021.

LIERLER, Y. 2022. An abstract view on optimizations in propositional frameworks. Unpublished draft available at https://arxiv.org/abs/2206.06440.

LIERLER, Y. AND SUSMAN, B. 2017. On relation between constraint answer set programming and satisfiability modulo theories. *Theory and Practice of Logic Programming*, *17*, 4, 559–590.

LIERLER, Y. AND TRUSZCZYNSKI, M. 2011. Transition systems for model generators — a unifying approach. *Theory and Practice of Logic Programming*, *11(4-5)*, 629–646. (Special Issue, Proceedings of the 27th International Conference on Logic Programming, ICLP 2011).

LIERLER, Y. AND TRUSZCZYNSKI, M. An abstract view on modularity in knowledge representation. In *Proceedings of the AAAI Conference on Artificial Intelligence* 2015.

LIFSCHITZ, V., TANG, L. R., AND TURNER, H. 1999. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, *25*, 369–389.

NIEUWENHUIS, R. AND OLIVERAS, A. On sat modulo theories and optimization problems. In BIERE, A. AND GOMES, C. P., editors, *Theory and Applications of Satisfiability Testing - SAT 2006* 2006, pp. 156–169, Berlin, Heidelberg. Springer Berlin Heidelberg.

OSTROWSKI, M. AND SCHAUB, T. 2012. Asp modulo csp: The clingcon system. *Theory and Practice of Logic Programming*, *12*, 4-5, 485–503.

PAPADIMITRIOU, C. AND STEIGLITZ, K. 1982. *Combinatorial Optimization: Algorithms and Complexity*, volume 32.

ROBINSON, N., GRETTON, C., PHAM, D.-N., AND SATTAR, A. Cost-optimal planning using weighted maxsat. In *ICAPS 2010 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling (COPLAS10)* 2010.

SEBASTIANI, R. AND TOMASI, S. Optimization in smt with la(q) cost functions. In GRAMLICH, B., MILLER, D., AND SATTLER, U., editors, *Automated Reasoning* 2012, pp. 484–498, Berlin, Heidelberg. Springer Berlin Heidelberg.

SHEN, D. AND LIERLER, Y. Smt-based constraint answer set solver ezsmt+ for non-tight programs. In *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR)* 2018.