

Techne: A(nother) Requirements Modeling Language

Alexander Borgida
Dept. of Computer Science
Rutgers University
borgida@cs.rutgers.edu

Alexei Lapouchnian
Dept. of Computer Science
University of Toronto
alexei@cs.toronto.edu

Neil Ernst
Dept. of Computer Science
University of Toronto
nernst@cs.toronto.edu

Sotirios Liaskos
School of Information Tech.
York University
liaskos@yorku.ca

Ivan J. Jureta
FNRS & Info. Management
University of Namur
ijureta@fundp.ac.be

John Mylopoulos
Dept. of Computer Science
University of Toronto
jm@cs.toronto.edu

Abstract

*This paper introduces the requirements modeling language called *Techne*, in response to a new core ontology for requirements [5]. The basic elements of *Techne* models are propositions that represent domain assumptions, goals, quality constraints, or tasks. Moreover, goals and quality constraints may be mandatory or optional and there can be preference relations defined over them. Given a requirements problem consisting of mandatory/optional goals and domain assumptions, a solution consists of a collection of tasks and quality constraints which together satisfy all mandatory goals and do as well as possible with respect to optional ones. A formal semantics is provided for *Techne*, along with a syntax, as well as illustrative examples.*

1 Introduction

Requirements modeling languages have been part of the very core of Requirements Engineering (RE) research since the days of SADT [9], with KAOS [3], *i** [11], and others, serving as state-of-the-art for much of the on-going research. Every requirements modeling language is grounded in an ontology of requirements, i.e., a set of assumptions about what requirements are. Traditionally, requirements were viewed as functions the system-to-be ought to support. This view is reflected in SADT and other structured analysis techniques of the '70s and '80s. More recently, an intentional perspective on requirements has gained ground: requirements are now viewed as stakeholder goals representing the intended purposes for the system-to-be. This deceptively small shift in the underlying ontology for requirements has had tremendous impact on requirements elicitation, modeling and analysis techniques.

Jureta et al. [5] proposed recently a new core ontology

for requirements which extends the goal-oriented perspective to allow for optional (“nice-to-have”) requirements and preferences (“requirement A is referred over requirement B”). The proposal also treats fully non-functional requirements (“softgoals”) in terms of approximations and quality constraints. In all, the proposal is a rather drastic extension of the small (and beautiful) set of concepts used in the past to formalize requirements (cf., e.g., [3, 12]). The extension, by the way, was much needed to align RE theory with RE practice where optionality and prioritization of requirements has been used routinely to manage requirements.

The main objective of this paper is to study the impact of the extended ontology for requirements (hereafter core ontology) on requirements modeling languages. We do so by proposing a new requirements modeling language called *Techne*, founded on the new core ontology. In addition, we adopt the definition of the requirements problem that accompanies the new core ontology and apply it to our language by defining precisely what does it mean for a specification (a set of tasks and associated quality constraints) to satisfy a given set of mandatory/optional requirements.

The rest of the paper is structured as follows. We first offer a nontechnical introduction to representation and reasoning in *Techne* via the standard meeting scheduler example (§2). We subsequently overview the syntax and semantics of *Techne* (§3) and discuss one approach to reasoning on *Techne* models (§4). We close with a discussion of related work, and a summary of conclusions and directions for future efforts (§5).

2 *Techne* Models

The purpose of *Techne* is to support the representation and reasoning about instances of the requirements problem and the alternative solutions to a given instance of the requirements problem. As the requirements problem is de-

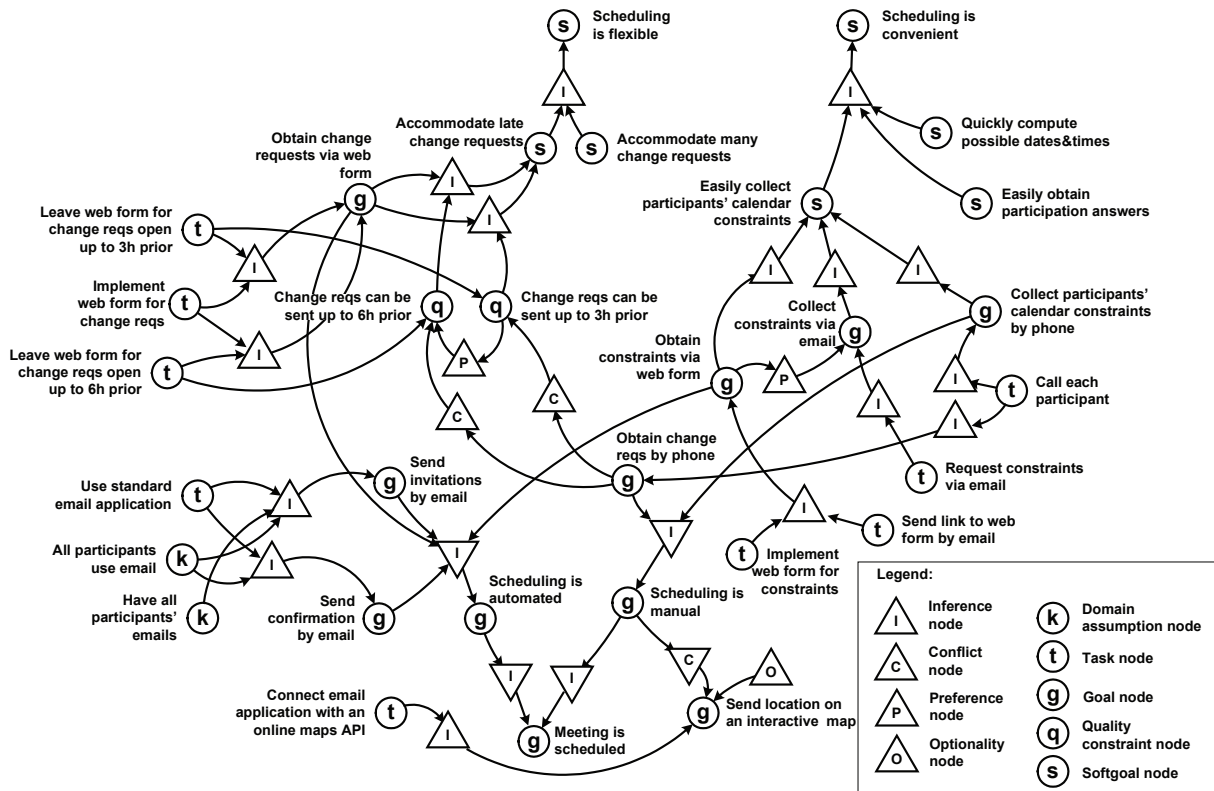


Figure 1. An r-net capturing requirements about the meeting scheduler problem.

defined via the concepts and relationships of the core ontology, Techne incorporates the *requirements net* (r-net), which captures the instances of the concepts and relationships of the core ontology. These instances are acquired via requirements elicitation, or produced by the application of RE methods. An r-net thereby represents the various goals, tasks, softgoals, and so on, along with specific relationships between these. Different parts of an r-net capture different solutions to the same given requirements problem. To facilitate the comparison of the solutions in terms of desirability, Techne uses the *solution net* (s-net); an s-net is obtained by a transformation of an r-net, and relates solutions in terms of preference and optionality. The aim in this section is to introduce the r-net and s-net via the standard meeting scheduler requirements problem, and in doing so recall the core ontology and highlight the salient features of the two Techne models.

Suppose that we need to build a meeting scheduler system. We can readily identify two ways, one automated and another manual to schedule a meeting. The former could involve the use of email and web forms. One web form could be used by participants to provide their calendar constraints, while another web form could be used to enter the requests to change the meeting date, time, and/or location. The links to the web forms and the invitations to meetings would be sent by email to all participants. The manual approach could simply amount to arrange a meeting with

participants via phone calls. If we consider in more detail each of these approaches, we can identify various functional and nonfunctional requirements for the meeting scheduler, along with tasks to perform in order to satisfy the various requirements, preferences between goals, tasks, and others, conflicts between some of them, and their refinements.

Concepts. An r-net for the meeting scheduler requirements problem is shown in Figure 1. An r-net is a directed labeled graph. Each node is a proposition about the system-to-be and/or its relevant environment. Figure 1 shows each proposition next to a node (circle or triangle), whereby each node is labeled by a symbol. The symbol indicates the concept or relationship of the core ontology, of which the given proposition is an instance. Following the core ontology, if a proposition represents a condition that is believed to hold about the system-to-be and/or its relevant environment, then this proposition is an instance of the **domain assumption** concept, and is labeled **k**. Desired conditions that should be satisfied are captured via instances of the **goal (g)**, **quality constraint (q)**, and **softgoal (s)** concepts. A goal will describe a verifiable functional condition (e.g., “**g**: Obtain change requests via web form” in Figure 1), while a quality constraint will further restrict the values of non-binary measurable characteristics of the system-to-be (e.g., “**q**: Change reqs can be sent up to 3h prior”). While a quality constraint will restrict the values of a quality defined over a well-defined quality space, a softgoal will do so over qual-

ities with ill-defined quality spaces (e.g., “s: *Accommodate late change requests*”). Tasks, i.e., instances of the **task** concept (**t**), capture the intentions to satisfy goals, quality constraints, and softgoals in some known manner: e.g., “t: *Use standard email application*” in Figure 1.

Relationships. Instances of domain assumptions give us instances of the relationships. E.g., if we believe that “satisfying the conjunction of two goals A and B is enough to satisfy the goal C”, then this is an instance of a domain assumption, which gives the relationship between the conjunction of the goals A and B, and the goal C. While not all domain assumptions convey relationships (e.g., “k: *All participants use email*” in Figure 1), we are interested in four relationships that some domain assumptions do convey: the binary inference (**I**), conflict (**C**), and preference (**P**), and the unary optionality (**O**). An instance of **inference** relates a set of instances of any concept or relationship to another instance of a concept or relationship, to convey that satisfying the conjunction of the former satisfies the latter. It is straightforward to see that an inference node can capture the usual (goal) refinement relationship (e.g., [3]), the means-ends and task decomposition relationships [11], and the approximation relationship, standing between a softgoal and its proxies (which are instances of other concepts and/or relationships in the core ontology) [5]. If two propositions/nodes are linked via a conflict node, then these propositions cannot be satisfied together.

Attitudes (i.e., emotions, feelings, and moods) are captured via instances of **optionality** and **preference** relationships. Optionality is an attribute of any of the said concepts indicating its optional or compulsory status: e.g., a candidate system-to-be must satisfy all compulsory goals, while it would be good if it satisfied as many of the optional goals as feasible. In Figure 1, the goal “g: *Send location on an interactive map*” is made optional by an optionality node connected to that goal. An instance of the preference relationship compares instances of any concept or relationship in terms of desirability: e.g., a preference will indicate that a goal is strictly preferred to another goal, that satisfying some goal is preferred to executing some plan, or that one preference is more important than another preference. Any preference node indicates strict preference: e.g., in Figure 1, a preference node indicates that the quality constraint “q: *Change reqs can be sent up to 3h prior*” is strictly preferred to the quality constraint “q: *Change reqs can be sent up to 6h prior*”.

R-nets and Usual Models. An r-net differs from the usual goal-oriented requirements models in several respects. (1) It can represent the instances of all concepts of the core ontology, along with the inference, conflict, preference, and optionality relationships. (2) An r-net is versatile: we can relate any node to any other node via an inference node, a conflict node, or preference node. We can therefore ex-

press preferences over preferences, preferences over inference applications, conflicts between preferences, conflicts between inference applications, and optional preferences, among others. (3) An r-net abstracts from the RE method used to obtain the content represented in that r-net. For example, we could have used the usual goal refinement method to conclude that the conjunction of the three goals “g: *Obtain change requests via web form*”, “g: *Send invitations by email*” and “g: *Obtain constraints via web form*”, refines the goal “g: *Scheduling is automated*”. Techne is only interested in how the former goals relate to the latter goal *in terms of satisfaction*, and is uninterested by what RE method was applied to obtain the goals, or to conclude the way ones affect the satisfaction of the others. In a summary then, Techne covers fully the core ontology, has a syntax that obeys less constraints than alternative goal-oriented modeling languages in RE (e.g., we can accommodate via inference nodes not only the goal refinement relationship of KAOS, but also the means-ends and task decomposition relationships of i^* , and the softgoal approximation relationship), and is not specialized and thereby constrained to particular RE methods.

Reasoning. A solution to the requirements problem is a combination of tasks and domain assumptions, such that the execution of the tasks in the conditions given by domain assumptions entails the satisfaction of all compulsory goals and quality constraints, and of zero or more optional goals and quality constraints. Given the r-net in Figure 1 the question to ask is if there are solutions in that r-net. If solutions are present, the next relevant question is how they compare in terms of preference and optionality. These questions are answered by the application of reasoning procedures on an r-net (cf., §4). To find a solution, we perform *simulation*: we assume that some sets of tasks are executed, and we traverse the r-net in order to determine the consequences of the execution of these tasks on the satisfaction of goals, quality constraints, and so on. E.g., if we assume that the task “t: *Use standard email application*” is executed (equivalently, satisfied or true), and that the domain assumptions “k: *All participants use email*” and “k: *Have all participants emails*” are true, then inference nodes tell us that the goals “g: *Send confirmation by email*” and “g: *Send invitations by email*” are satisfied/true. If we assume the truth of the task “t: *Call each participant*”, the goal “g: *Obtain change reqs by phone*” will be satisfied, but neither “q: *Change reqs can be sent up to 3h prior*” nor “q: *Change reqs can be sent up to 6h prior*” will be satisfied, since there the said goal is related to these two quality constraints via conflict nodes.

S-Net. The purpose of an s-net is to represent the comparison of solutions identified via simulation. If we perform simulations on the r-net in Figure 1, we will observe that some softgoals are not satisfied at all: e.g., “s: *Accommo-*

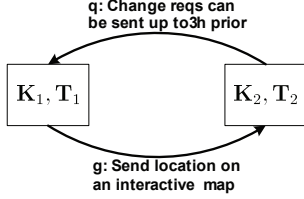


Figure 2. An s-net.

date many change requests” has no incoming lines at all, so that its satisfaction is unknown. This absence of information is not an uncommon situation, for an r-net is built iteratively. In order to accommodate incompleteness and inconsistency, we have four truth values in the Techne language: the usual **T** and **F**, for true (satisfied) and false (not satisfied) respectively, **U** for unknown, and **D** for disputed (i.e., some evidence concludes the satisfaction, some the denial of the node). We can still compare quasi-solutions in Figure 1 in an s-net. Suppose that we consider two quasi-solutions, say $(\mathbf{K}_1, \mathbf{T}_1)$ and $(\mathbf{K}_2, \mathbf{T}_2)$, from those in Figure 1, both involving automated scheduling. Suppose further that there are two differences between these two quasi-solutions: one is that $(\mathbf{K}_2, \mathbf{T}_2)$ satisfies the quality constraint “**q**: Change reqs can be sent up to 3h prior” and the $(\mathbf{K}_1, \mathbf{T}_1)$ instead satisfies “**q**: Change reqs can be sent up to 6h prior”, while the other difference is that the $(\mathbf{K}_1, \mathbf{T}_1)$ satisfies the optional goal “**g**: Send location on an interactive map” and $(\mathbf{K}_2, \mathbf{T}_2)$ does not satisfy that optional goal. The s-net that compares these two quasi-solutions is shown in Figure 2. A node in an s-net is a (quasi-)solution. A link goes from a solution A to another solution B iff either (i) A satisfies some optional node that B does not (in this case, the link is labeled with that optional node), or (ii) A satisfies a node, which is preferred to another node that B satisfies (in this case, the link is labeled with that preferred node). An s-net thereby summarizes the information relevant for the comparison of an (quasi-)solutions. Note that an s-net does not answer the question of which (quasi-)solution to choose among those available; instead, an s-net acts as a decision aid. In this sense, we see from Figure 2 that $(\mathbf{K}_1, \mathbf{T}_1)$ is a “better” quasi-solution than $(\mathbf{K}_2, \mathbf{T}_2)$ over one criterion (i.e., the quality constraint “**q**: Change reqs can be sent up to 3h prior”), while $(\mathbf{K}_2, \mathbf{T}_2)$ is better than $(\mathbf{K}_1, \mathbf{T}_1)$ over another criterion (i.e., the optional goal “**g**: Send location on an interactive map”). Which one will be chosen depends on the decision rule applied in a particular s-net; decision rules are not in the scope of Techne.

3 Language

Techne has two parts. The first part is *Techne logic* (TL), and is defined on a particular kind of r-nets, called attitude-free r-nets, and serves for the representation and identification (but not comparison) of solutions (cf., §3.1). The second part augments TL, in the sense that it enables reasoning

about how theories of TL compare over some specific criteria (cf., §3.2).

3.1 Techne Logic

Let p, q, r, s , indexed or primed when necessary, denote propositions, whereby a proposition is (as usual) a shareable content of beliefs, desires, or intentions, and the primary bearer of truth and falsity [10]. To remain general, we make no assumption on the formalism, in which a proposition is expressed: it is thereby unimportant if the proposition is in natural language, or is a well-formed formula with no free variables in some logic. A proposition p alone is not very interesting for the engineer, for it does not say whether the condition it describes is desired, believed, or otherwise, that is, if p is a goal, quality constraint, or otherwise. The elicitation of requirements provides a set of propositions P about the system-to-be and its relevant environment. The core ontology [5] provides a pragmatic characterization of the propositions: e.g., if p is believed by the stakeholders, then it is an instance of the domain assumption concept. The idea in Techne is to set each proposition $p \in P$ as a node in a graph, called *requirements net*, or r-net, whereby the pragmatic characterization is captured by assigning labels to the propositions/nodes. An r-net is a directed labeled graph, in which any two nodes can be connected by at most one link. Each proposition consequently carries one of the labels shown in the legend in Figure 1, whereby a label is intended to symbolize the concept or relationship that the given node instantiates.

We can obtain from any r-net R , the *attitude-free r-net* \bar{R} by removing all preference and optionality nodes, along with all links of R connected to these preference and optionality nodes. An \bar{R} thereby carries no information that allows the comparison of solutions, but still allows us to represent the instances of all concepts of the core ontology, along with the inference and conflict relationships between these instances.

The purpose of TL is to provide formal semantics to the nodes and links in attitude-free r-nets. We can otherwise perform no meaningful simulation on attitude-free r-nets in the aim of identifying solutions. To achieve this aim, the first step amounts to map the syntactic atoms of attitude-free r-nets to a symbolic syntax, which has standard negation and conjunction connectives, from which we define disjunction and material implication as abbreviations in the usual way. We can subsequently provide formal semantics (i.e., truth values) to well-formed formulas in TL in a standard way, and compute truth values via truth tables. Since we are not interested only in truth and falsity as usual, but also wish to accommodate incomplete information and inconsistency (cf., §2), we will have four possible truth values: $\{\mathbf{T}, \mathbf{F}, \mathbf{D}, \mathbf{U}\}$.

Syntax of \bar{R} . We distinguish three kinds of syntax for

attitude-free r-nets. The *visual syntax* is the syntax used to draw Techne models, such as the one shown in Figure 1; its syntactic elements are graphical primitives (circles, triangles, and lines with arrows) and letters used to label these primitives. In the *graph syntax*, and for a set of propositions P , any instance of a concept of the core ontology is written $\mathbf{x}(p)$ (with $p \in P$), where $\mathbf{x} \in \{\mathbf{k}, \mathbf{t}, \mathbf{g}, \mathbf{q}, \mathbf{s}\}$ are symbols for, respectively, an instance of the **domain assumption**, **task**, **goal**, **quality constraint**, and **softgoal** concepts. The relationship nodes written in graph syntax make explicit the nodes that they relate. E.g., the domain assumption $\mathbf{k}(q) = \text{“the conjunction of goals } \mathbf{g}(p_1), \dots, \mathbf{g}(p_n) \text{ refines the goal } \mathbf{g}(p)\text{”}$ gives an inference node, which we write $\mathbf{I}(q, \{\mathbf{g}(p_1), \dots, \mathbf{g}(p_n)\}, \mathbf{g}(p))$. E.g., the domain assumption $\mathbf{k}(q) = \text{“the conjunction of goals } \mathbf{g}(p_1), \dots, \mathbf{g}(p_n) \text{ is in conflict with the goal } \mathbf{g}(p)\text{”}$ gives a conflict node $\mathbf{C}(q, \{\mathbf{g}(p_1), \dots, \mathbf{g}(p_n)\}, \mathbf{g}(p))$. Finally, we have the symbolic syntax, which rewrites the visual and graph expressions as well-formed formulas of a propositional logic. Each of the first three columns in Table 1 provides the rules for generating valid expressions in each kind of syntax, and the correspondence between the elements of each kind of syntax. Valid expressions are generated inductively as usual, starting in each column from the first and moving to the last row.

An \bar{R} obeys a set of syntactic constraints, which are apparent from the inductive definition of (any of the three kinds of) syntax in Table 1. E.g., we cannot have a line between two instances of concepts (i.e., two nodes taking labels from the set $\{\mathbf{k}, \mathbf{t}, \mathbf{g}, \mathbf{q}, \mathbf{s}\}$). This is due to the need to make explicit the relationship between the instances of concepts, so that in a connected \bar{R} , we need to have at least an inference or conflict node to relate nodes that represent instances of concepts.

Two observations are in order regarding the move from visual/graph to symbolic syntax. (1) An inference node is rewritten as the implication of the conclusion node from the conjunction of the premise nodes, while the conflict node amounts to the implication of a negation from the conjunction of the attacking nodes. Relying on material implication is not the best choice for Techne, but is the only choice that would not violate the space constraints on the present discussion. Material implication suffers from well-known paradoxes, and is to be replaced in a future variant of Techne. (2) Several inference nodes that have the same conclusion are considered as alternatives, and are placed in disjunction. The rationale is that any, some, or all of the implications and their assumptions (i.e., x is called an assumption in $x \rightarrow y$) need to verify to reach the conclusion. If there are several conflict nodes that attack the same node, these nodes are also related via disjunction. Again, the rationale is that any of the conflict nodes and their premises need to hold for the target to be negated.

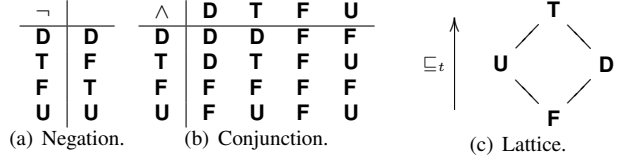


Figure 3. Truth tables and the lattice.

Semantics of \bar{R} . The syntactic elements of the symbolic syntax obtain semantics through Techne structures. A *Techne structure* is a tuple $(\mathcal{T}, \mathcal{D})$, where \mathcal{T} maps a syntactic element to a truth value. The codomain of \mathcal{T} includes four truth values taken from Belnap’s four valued logic [1]: $\mathbf{T}, \mathbf{F}, \mathbf{D}$, and \mathbf{U} correspond to, respectively *known-only-true*, *known-only-false*, *known-both-true-and-false*, and *unknown* truth values. \mathcal{D} maps a syntactic element to an instance of a concept or relationship of Techne (cf., §2; the codomain includes all possible instances of all concepts (i.e., domain assumption, task, goal, quality constraint, softgoal) and relationships (i.e., inference and conflict) that we wish to represent in attitude-free r-nets. Entailment in Techne, denoted $\models_{\mathcal{T}}$ is defined inductively via Techne structures in the last column of Table 1.






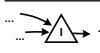
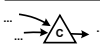

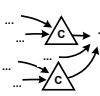
We define truth tables for negation (cf., Figure 3(a)) and conjunction (cf., Figure 3(b)) via the lattice in Figure 3(c). That it is a *lattice* means that it is a partial order \sqsubseteq_t , for which a unique greatest lower bound (meet) and least upper bound (join), denoted respectively $x \sqcap_t y$ and $x \sqcup_t y$, exist for each pair (x, y) of members of the ordered set of truth values. The lattice shown in Figure 3(c) is *complete*, meaning that it includes a meet and join for any subset of truth values (cf., e.g., [4]). Intuitively, the lattice in Figure 3(c) indicates the direction, in which truth “increases” (from bottom to top in Figure 3(c)). The four truth values “increase” from \mathbf{F} , over \mathbf{U} and \mathbf{D} , to \mathbf{T} . The truth tables for conjunction and disjunction follow immediately from the order defined via \sqsubseteq_t , by taking that meet \sqcap_t stands for conjunction \wedge and join \sqcup_t for disjunction \vee .

This leads us the notion of semantic entailment of a well-formed formula from another well-formed formula in TL; or, by equivalence of syntactic elements (cf., Table 1), the semantic entailment of a subgraph of an attitude-free r-net from another subgraph of the attitude-free r-net.

Definition 3.1. R-(Sub)Net Entailment. Let Ψ and Ψ' be two well-formed formulas in TL. We say that Ψ entails Ψ' , and write $\Psi \models_{\mathcal{T}} \Psi'$, iff for all Techne structures $(\mathcal{T}, \mathcal{D})$, we have $\mathcal{T}(\Psi) \sqsubseteq_t \mathcal{T}(\Psi')$.

Semantic entailment in TL corresponds to semantic entailment in Belnap’s four valued logic. While TL takes truth values and truth tables from Belnap’s logic, it remains original in the definition of syntax and Techne structures, and in the mapping from the syntactic elements to the instances of the concepts and relationships of the core ontology.

Table 1. Visual, graph, and symbolic syntax, and the semantics of attitude-free r-nets in Techne.

Visual syntax:	Well-formed subgraph Ψ in graph syntax:	Well-formed formula Ψ in symbolic syntax:	Semantics:
	$\mathbf{k}(p)$, i.e., a domain assumption node;	$\mathbf{k}(p)$	$(\mathcal{T}, \mathcal{D}) \models_T \mathbf{k}(p)$ iff $\mathcal{T}(\mathbf{k}(p)) = \mathbf{T}$ and $\mathcal{D}(\mathbf{k}(p))$ is an instance of the domain assumption concept;
	$\mathbf{t}(p)$, i.e., a task node;	$\mathbf{t}(p)$	$(\mathcal{T}, \mathcal{D}) \models_T \mathbf{t}(p)$ iff $\mathcal{T}(\mathbf{t}(p)) = \mathbf{T}$ and $\mathcal{D}(\mathbf{t}(p))$ is an instance of the task concept;
	$\mathbf{g}(p)$, i.e., a goal node;	$\mathbf{g}(p)$	$(\mathcal{T}, \mathcal{D}) \models_T \mathbf{g}(p)$ iff $\mathcal{T}(\mathbf{g}(p)) = \mathbf{T}$ and $\mathcal{D}(\mathbf{g}(p))$ is an instance of the goal concept;
	$\mathbf{q}(p)$, i.e., a quality constraint node;	$\mathbf{q}(p)$	$(\mathcal{T}, \mathcal{D}) \models_T \mathbf{q}(p)$ iff $\mathcal{T}(\mathbf{q}(p)) = \mathbf{T}$ and $\mathcal{D}(\mathbf{q}(p))$ is an instance of the quality assumption concept;
	$\mathbf{s}(p)$, i.e., a softgoal node;	$\mathbf{s}(p)$	$(\mathcal{T}, \mathcal{D}) \models_T \mathbf{s}(p)$ iff $\mathcal{T}(\mathbf{s}(p)) = \mathbf{T}$ and $\mathcal{D}(\mathbf{s}(p))$ is an instance of the softgoal concept;
	$\mathbf{I}(p, \{\Psi_i \mid 1 \leq i \leq m-1\}, \Psi_m)$, i.e., an inference node;	$(\bigwedge_{1 \leq i \leq m-1} \Psi_i) \rightarrow \Psi_m$	$(\mathcal{T}, \mathcal{D}) \models_T (\bigwedge_{1 \leq i \leq m-1} \Psi_i) \rightarrow \Psi_m$ iff $\mathcal{T}((\bigwedge_{1 \leq i \leq m-1} \Psi_i) \rightarrow \Psi_m) = \mathbf{T}$ and $\mathcal{D}((\bigwedge_{1 \leq i \leq m-1} \Psi_i) \rightarrow \Psi_m)$ is an instance of the inference rule relationship;
	$\mathbf{C}(p, \{\Psi_i \mid 1 \leq i \leq m-1\}, \Psi_m)$, i.e., a conflict node;	$(\bigwedge_{1 \leq i \leq m-1} \Psi_i) \rightarrow \neg \Psi_m$	$(\mathcal{T}, \mathcal{D}) \models_T (\bigwedge_{1 \leq i \leq m-1} \Psi_i) \rightarrow \neg \Psi_m$ iff $\mathcal{T}((\bigwedge_{1 \leq i \leq m-1} \Psi_i) \rightarrow \neg \Psi_m) = \mathbf{T}$ and $\mathcal{D}((\bigwedge_{1 \leq i \leq m-1} \Psi_i) \rightarrow \neg \Psi_m)$ is an instance of the conflict rule relationship;
	$\{\mathbf{I}(p_j, \{\Psi_{i,j} \mid 1 \leq i \leq m\}, \Psi) \mid 2 \leq j \leq n\}$, i.e., a set of inference nodes concluding the same node;	$\bigvee_{2 \leq j \leq n} ((\bigwedge_{1 \leq i \leq m} \Psi_{i,j}) \rightarrow \Psi)$	$(\mathcal{T}, \mathcal{D}) \models_T \bigvee_{2 \leq j \leq n} ((\bigwedge_{1 \leq i \leq m} \Psi_{i,j}) \rightarrow \Psi)$ iff $\mathcal{T}(\bigvee_{2 \leq j \leq n} ((\bigwedge_{1 \leq i \leq m} \Psi_{i,j}) \rightarrow \Psi)) = \mathbf{T}$ and for each $2 \leq j \leq n$, $\mathcal{D}((\bigwedge_{1 \leq i \leq m} \Psi_{i,j}) \rightarrow \Psi)$ is an instance of the inference rule relationship;
	$\{\mathbf{C}(p_j, \{\Psi_{i,j} \mid 1 \leq i \leq m\}, \Psi) \mid 2 \leq j \leq n\}$, i.e., a set of conflict nodes attacking the same node;	$\bigvee_{2 \leq j \leq n} ((\bigwedge_{1 \leq i \leq m} \Psi_{i,j}) \rightarrow \neg \Psi)$	$(\mathcal{T}, \mathcal{D}) \models_T \bigvee_{2 \leq j \leq n} ((\bigwedge_{1 \leq i \leq m} \Psi_{i,j}) \rightarrow \neg \Psi)$ iff $\mathcal{T}(\bigvee_{2 \leq j \leq n} ((\bigwedge_{1 \leq i \leq m} \Psi_{i,j}) \rightarrow \neg \Psi)) = \mathbf{T}$ and for each $2 \leq j \leq n$, $\mathcal{D}((\bigwedge_{1 \leq i \leq m} \Psi_{i,j}) \rightarrow \neg \Psi)$ is an instance of the conflict rule relationship;
	$\{\Psi_i \mid 1 \leq i \leq m\}$, i.e., a set of well-formed subgraphs;	$\bigwedge_{1 \leq i \leq m} \Psi_i$	$(\mathcal{T}, \mathcal{D}) \models_T \bigwedge_{1 \leq i \leq m} \Psi_i$ iff $\mathcal{T}(\bigwedge_{1 \leq i \leq m} \Psi_i) = \mathbf{T}$ and for each $1 \leq i \leq m$, $\mathcal{D}(\Psi_i)$ is an instance of a concept determined according to the rules above;
The following also apply:		$(\mathcal{T}, \mathcal{D}) \models_T \Psi_i \wedge \Psi_j$ iff $(\mathcal{T}, \mathcal{D}) \models_T \Psi_i$ and $(\mathcal{T}, \mathcal{D}) \models_T \Psi_j$; $(\mathcal{T}, \mathcal{D}) \models_T \neg \Psi$ iff $(\mathcal{T}, \mathcal{D}) \not\models_T \Psi$; $\Psi_j \rightarrow \Psi_j$ abbreviates $\neg \Psi_i \vee \Psi_j$; and $\Psi_j \vee \Psi_j$ abbreviates $\neg(\neg \Psi_i \wedge \neg \Psi_j)$.	

3.2 Comparison of Solutions

Comparing solutions requires a precise characterization of what a solution is in an attitude-free r-net \bar{R} in TL, and how preference and optionality relate solutions in an r-net R . To define the *solution* concept, we use particular kinds of subgraphs of \bar{R} or R . Namely, a **G-subnet** $\mathbf{G} = (V(\mathbf{G}), L(\mathbf{G}))$ is a subgraph of an r-net R , i.e., $V(\mathbf{G}) \subseteq V(R)$ and $L(\mathbf{G}) \subseteq L(R)$, such that: (1) $V(\mathbf{G})$ partitions onto (a) the set of all goals from R , and (b) the set of all domain assumptions from R , each of which relates at least one goal to at least another goal in R ; and (2) $L(\mathbf{G})$ contains all links from R that connect in R any two members of $V(\mathbf{G})$. With this in mind, it is straightforward to define **K**-, **T**-, **Q**-, and **S**-subnets, so that we omit these definitions. The purpose of each of these subnets is to group all instances of specific concepts within a single graph. This

leads us to the definition of the *solution* concept.

Definition 3.2. Solution. In *Techne*, $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$ is a *solution of the requirements problem* iff:

1. \mathbf{G}^c and \mathbf{Q}^c are subnets of, resp., \mathbf{G} and \mathbf{Q} ; and
2. $\mathbf{K}_i, \mathbf{T}_i \models_T \mathbf{G}, \mathbf{Q}$.

Observe that a solution is by definition a subnet of an attitude-free r-net \bar{R} , and a subnet of the corresponding r-net R . Also, a solution is by definition an *admissible solution*, that is, one satisfying all the conditions that *must* be satisfied if the stakeholders are not to reject that solution. For our meeting scheduler requirements problem, a way to read some solution $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$ is that \mathbf{K} indicates assumptions about the conditions, in which the system-to-be operates (e.g., all participants use email), \mathbf{T} indicates what the system-to-be does within/to its environment (e.g., calendar

constraints are requested via email), \mathbf{G} highlights the stakeholders' goals that it satisfies (e.g., meeting is scheduled), and \mathbf{Q} qualifies the goal-satisfying behavior of the system-to-be (e.g., late change requests can be accommodated).

Suppose now that we have identified n solutions: $(\mathbf{K}_1, \mathbf{T}_1, \mathbf{G}, \mathbf{Q}), \dots, (\mathbf{K}_n, \mathbf{T}_n, \mathbf{G}, \mathbf{Q})$. These solutions differ in terms of how many of the optional goals and optional quality constraints they satisfy, and how the goals and quality constraints they satisfy relate to one another in terms of preference. Different Techne structures satisfy different solutions. We therefore need means to *compare* Techne structures, and thereby solutions, which leads to the notion of *substructure* in Techne.

Definition 3.3. Substructure. *Let $(\mathcal{T}, \mathcal{D})$ and $(\mathcal{T}', \mathcal{D})$ be two Techne structures for an attitude-free r-net \bar{R} . We say that $(\mathcal{T}', \mathcal{D})$ is a **substructure of $(\mathcal{T}, \mathcal{D})$ in ϵ** , and write*

$$(\mathcal{T}', \mathcal{D}) <_{\epsilon}^T (\mathcal{T}, \mathcal{D})$$

*iff (i) \mathcal{T} and \mathcal{T}' have the same domain, (ii) all syntactic elements of \bar{R} except ϵ obtain the same truth value from both \mathcal{T} and \mathcal{T}' , and (iii) $\mathcal{T}'(\epsilon) \sqsubseteq_t \mathcal{T}(\epsilon)$.*¹

The substructure concept allows us to define what it means for a solution to be strictly more desirable than another one.

Definition 3.4. Better Solution. *Let $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$ and $(\mathbf{K}_j, \mathbf{T}_j, \mathbf{G}, \mathbf{Q})$, with $i \neq j$, be two solutions. We say that the solution $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$ is **strictly more desirable than $(\mathbf{K}_j, \mathbf{T}_j, \mathbf{G}, \mathbf{Q})$ in ϵ** , and write*

$$(\mathbf{K}_j, \mathbf{T}_j, \mathbf{G}, \mathbf{Q}) <_{\epsilon}^T (\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$$

iff for any two Techne structures $(\mathcal{T}, \mathcal{D})$ and $(\mathcal{T}', \mathcal{D})$ such that $(\mathcal{T}, \mathcal{D}) \models_T \mathbf{K}_i, \mathbf{T}_i$ and $(\mathcal{T}', \mathcal{D}) \models_T \mathbf{K}_j, \mathbf{T}_j$, either of the two conditions below holds:

1. $(\mathcal{T}', \mathcal{D}) <_{\epsilon}^T (\mathcal{T}, \mathcal{D})$;
2. *there is a strict preference (e.g., $\mathbf{P}(q, \epsilon, \epsilon')$) in R for ϵ over ϵ' , $\mathcal{T}'(\epsilon) \sqsubseteq_t \mathcal{T}(\epsilon)$ and $\mathcal{T}(\epsilon') \sqsubseteq_t \mathcal{T}'(\epsilon')$.*

The first condition in Definition 3.4 will verify when an (compulsory or optional) ϵ obtains a “higher” truth value (in terms of the lattice of truth values – cf., Figure 3(c)) in the Techne structures of the solution $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$ compared to the truth value it obtains in the Techne structures of the solution $(\mathbf{K}_j, \mathbf{T}_j, \mathbf{G}, \mathbf{Q})$. Comparison in terms of optionality will thus be performed through the first condition. The second condition in Definition 3.4 allows us to perform comparisons in terms of preferences. That condition tells us to strictly prefer that of the two solutions, which assigns a “higher” truth value to the preferred syntactic element (above, ϵ) and a lower truth value to the less preferred

¹ $\mathcal{T}'(\epsilon) \sqsubseteq_t \mathcal{T}(\epsilon)$ is abbreviation for “ $\mathcal{T}'(\epsilon) \sqsubseteq_t \mathcal{T}(\epsilon)$ and not $\mathcal{T}(\epsilon) \sqsubseteq_t \mathcal{T}'(\epsilon)$ ”.

syntactic element (above, ϵ'), whereby \sqsubseteq_t orders the truth values according to the lattice (cf., Figure 3(c)). E.g., it follows then that we will strictly prefer a solution that assigns \mathbf{T} to ϵ and \mathbf{F} to ϵ' to another solution that does the reverse; also, we will strictly prefer a solution that assigns \mathbf{U} to ϵ and \mathbf{F} to ϵ' , to the one that assigns \mathbf{F} to ϵ and \mathbf{T} to ϵ' .

The relation $<_{\epsilon}^T$ is specific to ϵ , which makes ϵ a *criterion* of comparison. *To say that a solution is better than another one is specific to a chosen criterion.* Solutions are consequently compared using a family of irreflexive binary relations $(<_c^T)_{c \in \mathcal{C}}$, where \mathcal{C} is the set of criteria. The criteria are by definition elicited, since they are captured in an r-net. Members of \mathcal{C} may be optional elements, and individual elements, which are subject to preferences. Some of these elements are obtained by the approximation of softgoals, so that criteria contain proxies for softgoals. Solutions can compare on potentially many dimensions, the number of which is $|\mathcal{C}|$. E.g., while we may have $(\mathbf{K}_j, \mathbf{T}_j, \mathbf{G}, \mathbf{Q}) <_{\mathbf{g}(p_1)}^T (\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$, we could also have $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q}) <_{\mathbf{g}(p_2)}^T (\mathbf{K}_j, \mathbf{T}_j, \mathbf{G}, \mathbf{Q})$. $\mathbf{g}(p_2)$ and $\mathbf{g}(p_1)$ may be incomparable in terms of preference (i.e., it is not meaningful to give a preference between them), so that Techne does not give a general rule as to which of the two solutions to choose. It will be up to the stakeholders and the engineer to choose one of the two.

To facilitate the comparison of solutions, Techne uses solution nets. An s-net is a directed labeled graph, in which each node is a solution. Each criterion c from $(<_c^T)_{c \in \mathcal{C}}$ is the member of the set of labels of links in an s-net. If we have only two solutions, such that $(\mathbf{K}_j, \mathbf{T}_j, \mathbf{G}, \mathbf{Q}) <_{\mathbf{g}(p_1)}^T (\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$ and $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q}) <_{\mathbf{g}(p_2)}^T (\mathbf{K}_j, \mathbf{T}_j, \mathbf{G}, \mathbf{Q})$, then the corresponding s-net will have two nodes, one for each solution, and two links, one labeled $\mathbf{g}(p_1)$ that runs from $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$ to $(\mathbf{K}_j, \mathbf{T}_j, \mathbf{G}, \mathbf{Q})$, while the other link will be labeled $\mathbf{g}(p_2)$, and will run from $(\mathbf{K}_j, \mathbf{T}_j, \mathbf{G}, \mathbf{Q})$ to $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$. An s-net for the meeting scheduler problem is shown in Figure 2.

4 Reasoning on Techne Models

Simulation amounts to hypothesize the execution of a subnet of the \mathbf{T} -subnet in an attitude-free r-net, then determine the effects of the hypothetical executions on the truth value of goals, quality constraints, and so on. Simulation is used to determine if some given tuple $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$ is a solution. Simulation is performed on an attitude-free r-net \bar{R} and a tuple $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$, such that each of the subnets in the tuple is a subnet of \bar{R} .

For the simulation of $(\mathbf{K}_i, \mathbf{T}_i, \mathbf{G}, \mathbf{Q})$ in \bar{R} , the first step is to make two assumptions: (1) $\forall \epsilon \in \mathbf{K}_i, \mathcal{T}(\epsilon) = \mathbf{T}$, and (2) $\forall \epsilon \in \mathbf{T}_i, \mathcal{T}(\epsilon) = \mathbf{T}$. The second step is to compute the truth values of the other syntactic elements of \bar{R} other than \mathbf{K}_i and \mathbf{T}_i . This amounts to calling $\text{EVALUATE}(\bar{R}, \bar{r})$, where \bar{r} is composed exactly of the subnets \mathbf{K}_i and \mathbf{T}_i .

Given the truth values for a subnet \bar{r} of an attitude-free r-net \bar{R} , we define below the algorithm EVALUATE, which computes the truth values of the syntactic elements of \bar{R} that are not in \bar{r} . Starting from the nodes in \bar{r} , the algorithm traverses \bar{R} and computes the truth values of the visited nodes. For a given node $p \in V(\bar{R})$, the NODELABEL procedure computes the truth value of p by applying the truth tables on the truth values propagated to p over the links that end in p . We discuss the NODELABEL procedure first (cf., §4.1), and then EVALUATE (cf., §4.2).

4.1 Node Labeling

NODELABEL uses the functions *val*, *deleteDuplicates*, and *concl*. *val* maps a node either to a truth value, or the value N . $val(v \in V(\bar{R}))$ returns N iff v was not already processed by NODELABEL, or if $v \notin V(\bar{r})$ (i.e., its truth value was not assumed at the outset); otherwise, *val*(v) returns the truth value returned previously by NODELABEL, or the truth value assumed at the outset for v .

The function *deleteDuplicates* returns, for a given collection, the set of all distinct elements in the collection.

The function *concl* takes a pair of truth values, and returns a set (potentially a singleton) of truth values. *concl* is defined over the truth table for implication and returns the possible truth values for the conclusion of an implication.² Say that we have an implication $x \rightarrow y$, and that we know the truth value $val(x)$ of x and the truth value $val(x \rightarrow y)$ of the implication $x \rightarrow y$. $concl(val(x), val(x \rightarrow y))$ returns the truth values of y that are allowed by the truth table for implication: there we will have, $val(x)$ is a row head, $val(x \rightarrow y)$ is in at least one cell of the row of $val(x)$, so that $concl(val(x), val(x \rightarrow y))$ is at least one of the column heads. E.g., $concl(\mathbf{U}, \mathbf{T}) = \{\mathbf{D}, \mathbf{T}\}$; $concl(\mathbf{T}, \mathbf{F}) = \{\mathbf{T}\}$.

The functions \sqcap_t and \sqcup_t return, respectively the meet and join for a given set of truth values, according to the truth ordering defined by \sqsubseteq_t in Figure 3(c).

The procedure NODELABEL involves the collect and select steps.

Collect. The **for each** loop performs the collection of truth values, from which one will be subsequently selected. For a given node $v \in V(\bar{R})$, the **for each** loop considers each node $w \neq v$, $w \in V(\bar{R})$ such that there is a line $wv \in L(\bar{R})$, from w to v . If w is an inference node that concludes v , i.e., $w = \mathbf{I}(\cdot, Y, X)$ and $v \in X$, we need to distinguish two cases: (i) if we do not know the truth value $val(w)$ of w , then we assume that the truth value of w is \mathbf{U} and add \mathbf{U} to the collection $D_{\mathbf{I}}$; (ii) if we know the truth value $val(w)$, then $concl(val(\bigwedge_{\Psi \in Y} \Psi), val(w))$ gives the set of truth values for the implication w , and we cautiously take the meet of that set, i.e., $\sqcap_t concl(val(\bigwedge_{\Psi \in Y} \Psi), val(w))$,

²It is straightforward to build the truth table for implication from the Table 1 and Figure 3.

Algorithm 1 Node Labeling

Input: An attitude-free r-net \bar{R} and a node $v \in V(\bar{R})$;
Output: Exactly one of the members of $\mathfrak{T} = \{\mathbf{T}, \mathbf{T}, \mathbf{F}, \mathbf{D}, \mathbf{U}\}$;
1: **procedure** NODELABEL(\bar{R}, v)
 Collect values:
2: Empty the collections $D_{\mathbf{I}}$ and $D_{\mathbf{C}}$
3: Empty the sets $D_{\mathbf{I}}^u$ and $D_{\mathbf{C}}^u$
4: **for each** $w \in V(\bar{R})$ s.t. $\exists wv \in L(\bar{R})$ **do**
5: **if** $w = \mathbf{I}(\cdot, Y, X)$, $v \in X$, $val(w) = N$ **then**
6: Add \mathbf{U} to $D_{\mathbf{I}}$
7: **else if** $w = \mathbf{I}(\cdot, Y, X)$, $v \in X$, $val(w) \neq N$ **then**
8: Add $\sqcap_t concl(val(\bigwedge_{\Psi \in Y} \Psi), val(w))$ to $D_{\mathbf{I}}$
9: **else if** $w = \mathbf{C}(\cdot, Y, X)$, $v \in X$, $val(w) = N$ **then**
10: Add \mathbf{U} to $D_{\mathbf{C}}$
11: **else if** $w = \mathbf{C}(\cdot, Y, X)$, $v \in X$, $val(w) \neq N$ **then**
12: Add $\neg(\sqcap_t concl(val(\bigwedge_{\Psi \in Y} \Psi), val(w)))$ to $D_{\mathbf{C}}$
13: **end if**
14: **end for**
 Select a value:
15: $D_{\mathbf{I}}^u \leftarrow deleteDuplicates(D_{\mathbf{I}})$
16: $D_{\mathbf{C}}^u \leftarrow deleteDuplicates(D_{\mathbf{C}})$
17: **if** $D_{\mathbf{I}}^u \cup D_{\mathbf{C}}^u = \emptyset$ **then**
18: **if** $val(v) \neq N$ **then**
19: Return $val(v)$ and stop.
20: **else**
21: Return \mathbf{U} and stop.
22: **end if**
23: **else**
24: Return $\sqcap_t \{ \sqcup_t D_{\mathbf{I}}^u, \sqcup_t D_{\mathbf{C}}^u \}$ and stop.
25: **end if**
26: **end procedure**

according to the truth-partial ordering \sqsubseteq_t of the truth lattice, and add this to the collection $D_{\mathbf{I}}$. The collection $D_{\mathbf{I}}$ obtains the truth values resulting from all w that are inference nodes and conclude v . This same rationale applies if $w = \mathbf{C}(\cdot, Y, X)$ and $v \in X$. We also distinguish two cases: (i) if we do not know the truth value $val(w)$ of w , then we assume that the truth value of w is \mathbf{U} and add \mathbf{U} to the collection $D_{\mathbf{C}}$; (ii) if we know the truth value $val(w)$, then $concl(val(\bigwedge_{\Psi \in Y} \Psi), val(w))$ gives the set of truth values for the implication w , and we cautiously take the meet of that set, i.e., $\sqcap_t concl(val(\bigwedge_{\Psi \in Y} \Psi), val(w))$, and add its negation to the collection $D_{\mathbf{C}}$. The collection $D_{\mathbf{C}}$ obtains the truth values resulting from all w that are conflict nodes and attack v .

Select. The *deleteDuplicates* function is used to produce two sets: the sets $D_{\mathbf{I}}^u$ and $D_{\mathbf{C}}^u$ carry all distinct truth values from, respectively, $D_{\mathbf{I}}$ and $D_{\mathbf{C}}$. We then distinguish two cases. If $D_{\mathbf{I}}^u \cup D_{\mathbf{C}}^u$ is empty, then there are no inference nodes that conclude v and no conflict nodes that attack v . We therefore check if v already has a truth value (which happens if its truth value was assumed at the outset of the reasoning procedure): (i) if it does have a truth value, then this truth value is kept; (ii) if it does not have a truth value, we assign \mathbf{U} to it. Instead, if $D_{\mathbf{I}}^u \cup D_{\mathbf{C}}^u$ is not empty, we compute the joins of, respectively $D_{\mathbf{I}}^u$ and $D_{\mathbf{C}}^u$, and return the meet of these two.

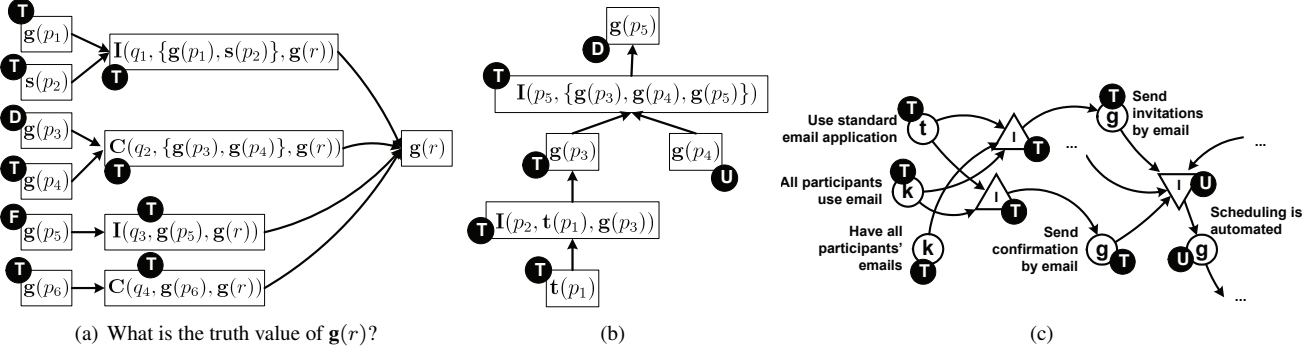


Figure 4. Illustration of node labeling and evaluation.

For illustration, and to see why we return the meet of the join of D_I^u and of the meet of D_C^u , consider the hypothetical attitude-free r-net written using graph syntax in Figure 4(a). The following are the computations performed by the procedure NODELABEL to determine the truth value of $g(r)$ in Figure 4(a):

- $\text{concl}(\text{val}(g(p_1) \wedge s(p_2)), \text{val}(I(q_1, \{g(p_1), s(p_2)\}, g(r)))) = \text{concl}(T, T) = \{T\}$;
- $\text{concl}(\text{val}(g(p_3) \wedge g(p_4)), \text{val}(C(q_2, \{g(p_3), g(p_4)\}, g(r)))) = \text{concl}(D, T) = \{T, U\}$;
- $\text{concl}(\text{val}(g(p_5)), \text{val}(I(q_3, g(p_5), g(r)))) = \text{concl}(F, T) = \{D, T, F, U, T\}$;
- $\text{concl}(\text{val}(g(p_6)), \text{val}(C(q_4, g(p_6), g(r)))) = \text{concl}(T, T) = \{T\}$;
- $\sqcap_t \{T\} = T$; $\neg(\sqcap_t \{T, U\}) = U$; $\sqcap_t \{D, T, F, U, T\} = F$;
- $\neg(\sqcap_t \{T\}) = F$;
- $D_I = (T, F)$; $D_C = (U, F)$; $D_I^u = \{T, F\}$; $D_C^u = \{U, F\}$;
- $\sqcap_t \{\sqcup_t D_I^u, \sqcup_t D_C^u\} = \sqcap_t \{T, F\} = F$;
- The truth value of $g(r)$ is F .

Proposition 4.1. Procedure NODELABEL applied to a node v of a finite attitude-free r-net \bar{R} (i) does not loop indefinitely, (ii) returns the truth value for v that is not different from a truth value that the truth tables allow us to compute for v , given the syntactic translation rules and the semantics in Table 1, and (iii) has the worst running time in $O(4!(\text{inDegree}(\bar{R}, v) + 2))$, where $\text{inDegree}(\bar{R}, v)$ is the number of links in \bar{R} that end in v .

Proof. Cf., Appendix A. \square

4.2 Evaluation

We start from an attitude-free r-net \bar{R} and its subnet \bar{r} . We assume a truth value for each node in $V(\bar{r})$. The aim is to determine the truth values of all nodes in $V(\bar{R})$. If we assume that \bar{R} is acyclical, doing this is straightforward, and formalized in Algorithm 2.³ It is not difficult to see that

³The presence of cycles gives rise to nontrivial complications. Two such complications are the possibility for truth values to be unstable, and potential absence of unique truth values. The identification of stable and unique truth values increases the complexity of the evaluation algorithm. We leave the treatment of cyclical attitude-free r-nets for future work.

Algorithm 2 Evaluation

Input: An attitude-free r-net \bar{R} , a subnet \bar{r} thereof, and a truth value for each node in the subnet \bar{r} ;

Output: For each node in $V(\bar{R})$, one truth value from $\{T, F, D, U\}$;

```

1: procedure EVALUATE( $\bar{R}, \bar{r}$ )
2:   Empty the queue  $Q$ 
3:   Add all nodes in  $V(\bar{r})$  to the queue  $Q$ 
4:   for each  $v$  in  $Q$  do
5:     for each  $w \in V(\bar{R})$  s.t.  $\exists vw \in L(\bar{R})$  do
6:       NODELABEL( $\bar{R}, w$ )
7:       Add  $w$  to  $Q$ 
8:     end for
9:   Delete  $v$  from  $Q$ 
10: end for
11: for each  $v \in V(\bar{R})$  s.t.  $\text{val}(v) = N$  do
12:   Assign  $U$  to  $v$ 
13: end for
14: end procedure

```

Algorithm 2 equates to a breadth first search (e.g., [6]), that is launched from each node in \bar{r} and traverses the graph over all outgoing links of a node. The second outer **for each** loop assigns the U truth value to all nodes that were not traversed by the first outer **for each** loop: any such vertex v has an unknown truth value, i.e., $(v) = N$.

Proposition 4.2. Procedure EVALUATE applied to a subnet \bar{r} of a finite and acyclic attitude-free r-net \bar{R} (i) does not loop indefinitely, (ii) for each node $v \in V(\bar{R})$, returns the truth value for v that is not different from a truth value that the truth tables allow us to compute for v , given the syntactic translation rules and the semantics in Table 1, and (iii) has the upper bound on the running time in $O(P_{\bar{r}}(|V(\bar{r})| + |L(\bar{R})|))$, where $P_{\bar{r}}$ is the number of simple paths in $V(\bar{R})$ such that each of these paths starts in a node of \bar{r} .

Proof. Cf., Appendix B. \square

We take two examples for illustration. Consider first the attitude-free r-net in Figure 4(b). Suppose that T is assigned at the outset to both $t(p_1)$ and $I(p_2, t(p_1), g(p_3))$. Since $I(p_2, t(p_1), g(p_3))$ is true, we know that the implication $t(p_1) \rightarrow g(p_3)$ is true. To determine the truth value of

$\mathbf{g}(p_3)$, we consult the truth table for implication, and conclude that $\mathbf{g}(p_3)$ receives the truth value \mathbf{T} . Since we lack information already to compute the truth value of $\mathbf{g}(p_4)$, we assume that its truth value is \mathbf{U} . By using the translation rules on $\mathbf{I}(p_5, \{\mathbf{g}(p_3), \mathbf{g}(p_4), \mathbf{g}(p_5)\})$, we have the implication $(\mathbf{g}(p_3) \wedge \mathbf{g}(p_4)) \rightarrow \mathbf{g}(p_5)$. Since $\mathbf{g}(p_3)$ is \mathbf{T} and $\mathbf{g}(p_4)$ is \mathbf{U} , the conjunction $\mathbf{g}(p_3) \wedge \mathbf{g}(p_4)$ is \mathbf{U} . We assumed at the outset that the implication $(\mathbf{g}(p_3) \wedge \mathbf{g}(p_4)) \rightarrow \mathbf{g}(p_5)$ is \mathbf{T} . With this in mind and from the truth table for implication, we see that $\mathbf{g}(p_5)$ can be any of $\{\mathbf{D}, \mathbf{T}\}$. We take the meet of this set, $\sqcap_t\{\mathbf{D}, \mathbf{T}\}$, which is \mathbf{D} according to the lattice in Figure 3(c). In the second example, shown in Figure 4(c) we have assumed the truth of the task, two domain assumptions, and two inferences on the left-hand side of the figure. This led us to label \mathbf{T} the goals “ \mathbf{g} : *Send invitations by email*” and “ \mathbf{g} : *Send confirmation by email*”. Without making additional assumptions, we can only conclude \mathbf{U} on the inference that concludes the goal “ \mathbf{g} : *Scheduling is automated*”, and subsequently \mathbf{U} on that same goal.

5 Conclusions and Future Work

We have proposed a novel requirements modeling language called *Techne*, summarized its logical semantics and presented basic algorithm for reasoning with *Techne* models. These models (r-nets) are directed labeled graphs with many-sorted nodes, some of which represent propositions such as goals, quality constraints, softgoals, tasks, while other propositions represent background knowledge, preferences and other logical relationships. These nodes are connected by simple edges, so that all aspects of the requirements are reified. A requirements problem is defined by an r-net along with information as to whether its goal/quality elements are mandatory (“must-have” requirements) or optional (“nice-to-have”). The semantics of *Techne* define solutions to a given requirements problem as a set of tasks and quality constraints which together satisfy all mandatory requirements and do as well as possible with respect to optional ones and preferences. In general, a given requirements problem will admit many such solutions, incomparable to each other.

Techne constitutes a significant departure from requirements modeling languages in the family of KAOS and i^* in a number of directions. First, the language is propositional, i.e., its primitive units are propositions, rather than offering an ontology of actors, actions, entities/relationships and the like. Second, the language takes a clear stand as to what constitutes a solution to a requirements problem. This has been an open issue with requirements languages that used softgoals and qualitative contributions from one requirement to another (whereby fulfillment of one requirement helps/hurts fulfillment of another). Indeed, *Techne* replaces such partial contributions with the notions of optional requirements, preferences and optimal fulfillment: an

admissible solution A is better than another B if it does better with respect to optional requirements and preferences (but both must satisfy mandatory ones). This means that in *Techne*, a requirements problem constitutes an optimization problem, rather than a satisfaction (or “satisficing”) one. Finally, *Techne* treats softgoals as first-class citizens along with goals, rather than as secondary criteria useful for comparing alternative solutions to goals.

The notion of optional requirements and preferences was inspired by earlier results in Knowledge Representation [2]. Liaskos [8] has already adopted such results to propose an expressive language for representing preferences within a goal-oriented framework. *Techne* is less expressive, computationally more tractable, and also offers a more seamless integration of requirements concepts included in the core ontology into a coherent modeling language.

Much remains to be done in turning *Techne* into a useful tool for requirements engineers. Firstly, we will extend the modeling framework to support social concepts, as provided in i^* . Secondly, we have come to appreciate the importance of contextual modeling, where, for instance, the requirements for a given sales system may be different depending on the type of customer or the size of the sale. Lapouchnian and Mylopoulos [7] make a proposal which we intend to incorporate in future versions of *Techne*. Finally, the reader may have noticed that there is no sublanguage within *Techne* for expressing quality constraints. We actually envision *Techne* being augmented with such a sublanguage that allow one to express quality constraints on the development process, run-time performance of the system-to-be, design artifacts (e.g., architecture) and more. We see such an extension as being defined by a collection of classes and associations (the things that can be talked about) along with a logical language in the family of OCL where one can express constraints such as “At least 95% of all sales process instances are completed”, or “the development process will last at most 6 months”.

A Proof of Proposition 4.1

Proof. (i) Termination. We first prove that NODELABEL applied to a node v of a finite attitude-free r-net \bar{R} (i) *does not loop indefinitely*. The **for each** loop considers once each node $w \neq v$ such that there is a link $wv \in L(\bar{R})$, from w to v . Since \bar{R} is finite, the **for each** loop always terminates. As the number of links ending in v is finite, so will be the size of D_{\perp} and $D_{\mathbf{c}}$, and consequently of D_{\perp}^u and $D_{\mathbf{c}}^u$. NODELABEL therefore never loops indefinitely for a finite \bar{R} .

(ii) Correctness. We prove that NODELABEL applied to a node v of a finite attitude-free r-net \bar{R} (ii) *returns the truth value for v that is not different from a truth value that the truth tables allow us to compute for v , given the syntactic*

translation rules and the semantics in Table 1. To prove this, it is enough to prove that NODELABEL computes the truth value of a node v by (A) taking into account all implications that conclude that node, (B) using truth tables of Techne to compute the truth of the conclusion of each implication, and (C) using translation rules from Table 1 to write the implications that it uses when computing the truth value of the conclusion of each implication. Proposition (A) is evident from the **for each** loop, which will traverse all nodes w such that there is a line $wv \in L(\bar{R})$, and thereby all inference and conflict nodes that, respectively, conclude and attack a given node v . Proposition (B) follows immediately from the definition of the functions val and $concl$, called in NODELABEL. That Proposition (C) is true can be seen from the **if** block in the **for each** loop, the way that NODELABEL uses the collections D_{\perp} and D_{C} and the sets D_{\perp}^u and D_{C}^u .

(iii) *Complexity.* Let $\mathfrak{F} = \{\mathbf{T}, \mathbf{F}, \mathbf{D}, \mathbf{U}\}$. We finally prove that NODELABEL applied to a node v of a finite attitude-free r-net \bar{R} (iii) has the worst running time in $O((|\mathfrak{F}| - 1)!(inDegree(\bar{R}, v) + 2))$, where $inDegree(\bar{R}, v)$ is the number of links in \bar{R} that end in v . The **for each** loop considers exactly once each node $w \neq v$ such that there is a link $wv \in L(\bar{R})$. We assume that evaluating $\bigwedge_{\Psi \in Y} \Psi$ is trivial. In the worst case, $concl(val(\bigwedge_{\Psi \in Y} \Psi), val(w)) = \mathfrak{F}$, and so for each considered w . Consequently, the **for each** loop runs in $O(inDegree(\bar{R}, v)(|\mathfrak{F}| - 1)!)$. Computing the meet (or join) for \mathfrak{F} takes $(|\mathfrak{F}| - 1)!$ since the meet (and join) is defined via a partial order, which is by definition reflexive, transitive, and anti-symmetric. The selection step involves the deletion of duplicates in D_{\perp} , D_{C} , and C , and the computation of the join for D_{\perp}^u , the meet for D_{C}^u , and of the meet of $\{\sqcup_t D_{\perp}^u, \sqcap_t D_{\text{C}}^u\}$. Assuming that the deletion of duplicates is trivial, computing $\sqcap_t \{\sqcup_t D_{\perp}^u, \sqcap_t D_{\text{C}}^u\}$ takes $2((|\mathfrak{F}| - 1)!)$ in the worst case, when $D_{\perp}^u = D_{\text{C}}^u = \mathfrak{F}$. It follows that NODELABEL has the worst running time in $O(inDegree(\bar{R}, v)(|\mathfrak{F}| - 1)! + 2(|\mathfrak{F}| - 1)!)$, that is, $O((|\mathfrak{F}| - 1)!(inDegree(\bar{R}, v) + 2))$ when applied to a node v of a finite \bar{R} . \square

B Proof of Proposition 4.2

Proof. (i) *Termination.* We first prove that EVALUATE applied to a subnet \bar{r} of a finite and acyclic attitude-free r-net \bar{R} (i) does not loop indefinitely. Since \bar{R} is acyclic, the number of times a node will be traversed by the first outer **for each** loop equals the number of simple paths from each member of $V(\bar{r})$. The number of simple paths in a finite acyclic directed graph is finite, so that each node will be traversed a finite number of times. At every traversal of a node, that node is removed from Q , so that Q will ultimately empty and the first outer **for each** loop will terminate. The second outer **for each** loop considers at most all nodes exactly once, so that it terminates as well. It follows that EVAL-

UATE will never loop indefinitely for an acyclic and finite \bar{R} .

(ii) *Correctness.* The first outer **for each** loop calls NODELABEL on each node it traverses, and each time it traverses that node. Observe that if there is a path from each of some two members w_1 and w_2 of $V(\bar{r})$ to a node v , and there is no path from another node in $V(\bar{r})$ to v , then $NODELABEL(\bar{R}, v)$ will be called exactly twice. If the path from w_1 to v is longer than the path from w_2 to v , then the first $NODELABEL(\bar{R}, v)$ call will be made when the first outer **for each** loop considers v along the shorter path. That first call will provide a *preliminary* truth value for v . This value is preliminary, since not all nodes on the loonger path will have been evaluated when the loop reaches v along the shorter path. The second $NODELABEL(\bar{R}, v)$ call will provide the definite truth value. Correctness of EVALUATE therefore depends on the correctness of NODELABEL. Following Proposition 4.1, EVALUATE is correct when applied to an acyclic and finite \bar{R} .

(iii) *Complexity.* It is straightforward to see that the number of times a node v will enter the queue Q equals the total number of simple paths to v from all members of $V(\bar{r})$. The upper bound on the running time is then in $O(P_{\bar{r}}(|V(\bar{r})| + |L(\bar{R})|))$, where $P_{\bar{r}}$ is the number of simple paths in $V(\bar{R})$ such that each of these paths starts in a node of \bar{r} . \square

References

- [1] N. D. Belnap, Jr. A useful four-valued logic. In J. M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*. D. Reidel Publishing Co., 1977.
- [2] M. Biennu, C. Fritz, and S. A. McIlraith. Planning with qualitative temporal preferences. In *Knowl. Rep. and Reasoning*, 2006.
- [3] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, 1993.
- [4] M. L. Ginsberg. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Comput. Intell.*, 4:265–316, 1988.
- [5] I. J. Jureta, J. Mylopoulos, and S. Faulkner. Revisiting the core ontology and problem in requirements engineering. In *16th IEEE Int. Requirements Engineering Conf.*, 2008.
- [6] D. E. Knuth. *The Art Of Computer Programming*, volume 1. Boston: Addison-Wesley, 3rd edition, 1997.
- [7] A. Lapouchnian and J. Mylopoulos. Modelling domain variability with contexts. Submitted for publication.
- [8] S. Liaskos. *Acquiring and Reasoning about Variability in Goal Models*. PhD thesis, Dept. Comput. Sci., University of Toronto, 2008.
- [9] D. A. Marca and C. L. McGowan. *SADT: structured analysis and design technique*. McGraw-Hill, Inc., 1987.
- [10] M. McGrath. Propositions. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008.
- [11] E. Yu. Towards modeling and reasoning support for early requirements engineering. In *Proceedings of the IEEE International Symposium on Requirements Engineering*, 1997.
- [12] P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM T. Softw. Eng. Methodol.*, 6(1):1–30, 1997.