Task Instruction

Shah, J.A.; Gluck, K.A.; Belpaeme, T.; Koedinger, K.R.; Rohlfing, K.J.; van der Maas, H.L.J.; Van Eecke, P.; VanLehn, K.; Vollmer, A.-L.; Yee-King, M.

DOI
10.7551/mitpress/11956.003.0016

Publication date
2018

Document Version
Final published version

Published in
Interactive Task Learning

Interactive Task Learning

Edited by Kevin A. Gluck and John E. Laird

MITCogNet
From the MIT Press

Series Editor: J. R. Lupp
Editorial Assistance: A. Ducey-Gessner, M. Turner, C. Stephen
Photographs: N. Miguletz
Lektorat: BerlinScienceWorks

The book was set in TimesNewRoman and Arial.
Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data is available

Ernst Strüngmann Forum (26th: 2017 : Frankfurt am Main, Germany)

ISBN: 978-0-262-03882-9

10 9 8 7 6 5 4 3 2 1

11

Task Instruction

Julie A. Shah, Kevin A. Gluck, Tony Belpaeme,
Kenneth R. Koedinger, Katharina J. Rohlfing,
Han L. J. van der Maas, Paul Van Eecke, Kurt VanLehn,
Anna-Lisa Vollmer, and Matthew Yee-King

Abstract

An early concept of interactive task learning (ITL) assumed a human teacher and machine learner. This book broadens the thinking about this relationship by explicitly allowing flexibility regarding the teacher and learner roles. Future ITL systems will be maximally useful and beneficial to the extent that they are effective and efficient *learners* as well as effective and efficient *instructors*. Focusing on task instruction, the primary goal of this chapter is to relate the critical role of instruction in ITL to key existing literature from related areas of research. The general concept of co-constructive task instruction is introduced and differentiated from traditional conceptualizations of fixed instructor and learner roles. Frameworks, models, and methods for task instruction are discussed, and broad connections are made between ITL and structural and adaptive improvements to instruction, historical developments in programming, and the extraordinary challenge that fluid, flexible, co-constructive task instruction and learning places on the vision for ITL.

Co-Constructive Task Instruction

The human and machine roles in interactive task learning (ITL) were viewed earlier as strictly discrete and fixed, with one always serving as the teacher and the other as the learner (Laird et al. 2017a). We anticipate it will be more accurate and useful to view these roles as fluid and dynamic, with the human and computer learning together and co-constructing an understanding of their task.

Group photos (top left to bottom right) Julie Shah, Kevin Gluck, Katharina Rohlfing, Matthew Yee-King, Anna-Lisa Vollmer, Tony Belpaeme, Han van der Maas, Ken Koedinger, Julie Shah, Paul Van Eecke, Katharina Rohlfing, Anna-Lisa Vollmer, Han van der Maas, Matthew Yee-King, Paul Van Eecke, Kurt VanLehn, Tony Belpaeme, Ken Koedinger, Julie Shah, Kurt VanLehn, Kevin Gluck

In co-construction, the aspect of iteration is crucial for the model of the learner and teacher because it changes the notion of intent (Figure 11.1). Accordingly, when the course of interaction is viewed as a possible method of providing feedback and instruction (Figure 11.1a), the initial intent remains stable throughout the entire interaction and is generated by the teacher as s/he guides the learner.

However, the iterative nature of an interaction offers not only the possibility of giving feedback or instruction, it also provides the possibility to adjust and align with the learner's action. In this way, it is possible for the initial intent to be changed or reformulated as the interaction unfolds. Teaching within this framework refers to a sequence of coactions established to achieve a specific goal. A teacher will have knowledge of the tasks that must be completed to arrive jointly at the goal. A knowledgeable teacher is able to constrain the actions of a learner such that the intended goal will be reached (Heller and Rohlfing 2017). However, in this framework, the strict role division between learner and teacher vanishes: individual actions that must be performed can be taken by both partners as they each contribute to reaching the joint goal.

The process of *co-construction* is temporal and embeds learning and future interaction for both human and machine. Co-construction is a process through which mutual understanding or "common ground" can be achieved between

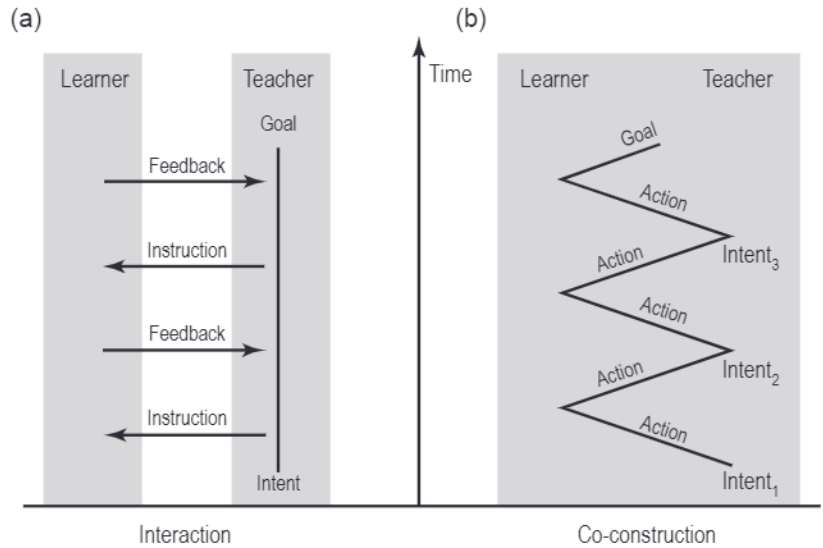

Figure 11.1 The iterative nature of an interaction is depicted: (a) the teacher has an intent, provides instructions to the learner accordingly, and receives feedback; (b) both agents co-construct a joint goal. While in the approach focusing on the feedback (a), roles of the agents are separated, they are both responsible for the intent in the co-constructing approach (b).

the teacher and learner. Common ground, in this sense, is neither fixed nor established once in an interaction, but is subject to constant change and evolution. Importantly, it is based on the history of interaction.

At the beginning of the learning process, the language channel between human and machine (gray arrows, Figure 11.2) may be quite limited. Nevertheless, the agents can make progress through ITL via shared interactions in the world, as when the teacher demonstrates, the learner practices, and the teacher adjusts to the learner's multimodal action and reformulates the goal(s) for their joint actions. Both agents may make coordinated use of language, gestures, and actions which, while not immediately understood, embed the understanding of language into shared actions toward specific goals (bottom arcs).

As the learning process progresses, the agents become increasingly experienced in working together and adjust to one another. As a consequence of relying on their joint experiences in perceiving and manipulating the world/task environment (Figure 11.2, right bottom arcs), the two partners develop new, more conventionalized forms of language interaction (represented by the thicker arcs at the upper right, Figure 11.2). Task- and communication-relevant knowledge inside the machine and human also grow (not shown).

Team training practices can facilitate and accelerate the process of humans and robots learning to work together (Gorman et al. 2010; Nikolaidis et al. 2015; Ramakrishnan et al. 2017). The cross-training approach is designed to improve team adaptivity through practice by requiring team members to switch roles with one another. Through this process, teammates are able to build a shared mental model of the task and collaboration strategy, and they can thus better anticipate one another's actions (Nikolaidis et al. 2015). In another team training method, perturbation training, a team experiences slight disturbances (perturbations) during the training process that are intended to help team members learn to coordinate effectively under new variants of the given task. This method is well-suited to heterogeneous teams, as each member practices his

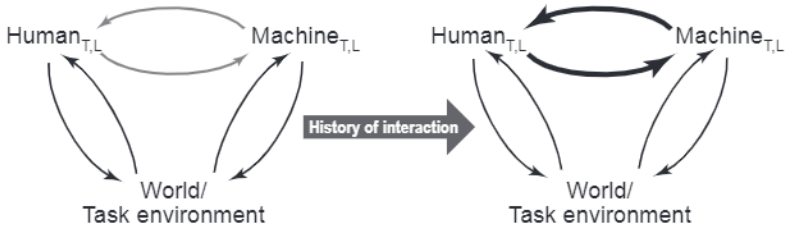

Figure 11.2 For human and machine, co-construction embeds learning and future interaction. Agents cannot rely on common knowledge or language; they must demonstrate individual understanding through actions (left: stronger links to world/task environment are necessary). With recurrent interaction, agents can increasingly rely upon joint experiences, grounded with language (right: stronger links between agents indicate increasing knowledge of each other).

or her own role on the team during the training process; it also does not require switching roles among team members, as in the cross-training method. Results from recent human studies (Gorman et al. 2010) and human–robot studies (Ramakrishnan et al. 2017) indicate that perturbation training yields high levels of human–human and human–robot performance on novel tasks.

Illustrative Example for ITL

In human–robot interaction, many factors have an impact on teaching. The human teacher's expectations for and knowledge of the robot learner are shaped by the robot's appearance, (mis)conceptions acquired from media and general knowledge, any prior human–robot interactions, and the *in situ* behavior of the robot (Hegel et al. 2009; Pitsch et al. 2013; Vollmer et al. 2014). In humans, a basic understanding of the learner can be derived from their outward appearance. For example, we infer that infants do not have the verbal capabilities of adults. By contrast, software agents often have no physical experience, and two robots with the same appearance are not necessarily equipped with identical software and capabilities. Thus, during interactions in which a machine learns from a human user, part of the student model that we normally would assume is available to the teacher (i.e., its existing knowledge and skills) may be unknown to its human instructor.

Consider a scenario in which a robot learner is being taught to clean windows by a human. The robot learner is a full-size humanoid robot with the following characteristics:

- It can perceive 3D motions of manipulative action demonstrations with sensors/cameras.
- It knows that both end-effector position and trajectory may be important for goal attainment.
- It understands the concept of "careful" performance (with regard to the amount of force applied).
- It is able to generalize a shown movement to a certain extent, such that movements performed with one object can be carried out with different objects.

The co-construction interaction unfolds as follows:

Teacher: Takes the cleaning agent, and says, "Look, I will show you how to clean this window now. You have to be really careful; it is quite old." T sprays the cleaning agent on the window.

Teacher: Takes the orange cloth, places it on the window, and says: "It's important not to push against the window too hard," and moves the cloth in straight lines from top to bottom. At the end of the

demonstration, T moves the cloth around the edge of the window and says, "Done. Do you want to try it?" T puts the cloth on the windowsill and begins to spray the cleaner on the window again.

Learner: Says, "Okay," attempts to grasp the cloth, and fails. (End effector not suitable to grasp this object).

Teacher: After observing two failed attempts, T says, "Oh, sorry. Don't worry. You can use the sponge as well," and gives the robot the blue sponge.

Learner: Moves its arm and places the sponge gently onto the window, removes it again, then sets it down on the windowsill at the exact place the teacher put the cloth.

Teacher: Says, "Oh no, no, It's about the movement I showed. I'll show it again." T picks up the sponge and demonstrates the downward motion again on the window (a simpler demonstration: slower, with, more pauses, while monitoring the learner's attention), this time with the sponge. T sets the sponge down on the windowsill and sprays the window again.

Learner: Says, "I think I've got it now," picks up the sponge, places it gently on the window and performs a few up-and-down movements, then sets the sponge down on the window sill. "Done."

Teacher: Says, "Great, you repeat these movements until there is no cleaning agent on the glass anymore. This is how you clean a window."

Learner: Says, "Got it."

Co-Constructing Intentions

The co-constructive view of learning has certain implications for intentions that are usually considered to drive the partners' actions. In a co-construction process, partners interact iteratively, in turns, to reach joint agreement on their intentions and goals; consequently, modifications to movement and intent occur as the process unfolds. Offering the learner a different object with which to perform the task following multiple failed attempts is an instance of such adaptation. It requires the teacher to have perceived prior failed attempts and understand what the problem is.

In the window-cleaning example, the fact that the robot's gripper (end effector) is not suitable to grasp the cloth represents knowledge about the learner. This could be a part of the domain knowledge that is initially unknown to both the teacher and learner, which they will discover together over the course of the interaction. As another example of domain knowledge, consider a scenario in which the robot has learned about the goal state of the window-cleaning task and, during its turn, discovers a spot on the window that will not come off when cleaned using the sponge. The robot then suggests, "There is a spot of what I think is paint. I cannot remove it with the sponge. We need a different

tool." At this moment, the robot leads the interaction and communicates novel information to the human, who, accordingly, learns about the spot from the robot. In such a case, task knowledge has changed: it now concerns the removal of paint from the glass. The tutor might then show the robot how to remove the spot of paint with a razor blade.

This illustrates that intentions are generated over the course of interaction as the domain knowledge evolves for the partners.

Jointly Clarifying and Completing the Goal

Similar to the dynamic view of intentions, the co-constructive view of learning implies that partners are working jointly toward a goal. In the window-cleaning task above, when the teacher cleans the window for a second time, the teacher monitors the learner more closely over the course of the demonstration. If the teacher observes, at the beginning, that the robot's eye gaze is shifting from the relevant object (the sponge) to the goal position on the windowsill, the teacher might stop the movement, shake the object, or say "Robot! Look!" to regain the robot's attention. Once the robot's gaze is again on the object, the teacher's movements become exaggerated to ensure that the robot continues to follow (track) the movement.

This interaction demonstrates that human teachers are very flexible; they adapt and adjust dynamically to the learner, taking the history of their interaction into account. In addition, both the teacher and learner adapt to the environment. For instance, the teacher might intend to produce the same movement with the sponge as with the cloth, but during the production/demonstration of the movement, the teacher observes that the sponge covers a larger area of the window and that fewer up and down movements are necessary. Consequently the teacher will produce the movements further apart from one another, deviating from the spacing of movements when using the cloth. In addition, the teacher has a certain goal intent and an intended action production (i.e., demonstration), but the realized movement may not coincide with either of these initial intents.

In the above case, the final production of the robot is actually not satisfying the requirements of the goal state. The teacher, thus, understands that the robot learner has not yet understood the goal state and clarifies the goal verbally.

If the modality of language is unavailable (e.g., if the robot simply lacks speech recognition, or there is a construction site outside the window and speech recognition is difficult in noisy environments), or the robot simply does not know what it has to do to remove all of the cleaning agent from the window, the teacher might include an additional iteration at the end of the interaction, modifying movement and intent to emphasize removal of the cleaning agent.

Frameworks for Task Instruction

ICAP Framework

The interactive, constructive, active, and passive (ICAP) hypothesis predicts the effectiveness of various types of instruction (Chi 2009; Menekse et al. 2013; Chi and Wylie 2014a; Chi and Menekse 2015). It defines four classifications of observable student behavior:

- *Interactive:* "We operationalize *interactive* behaviors to dialogues that meet two criteria: (a) both partners' utterances must be primarily *constructive*, and (b) a sufficient degree of turn-taking must occur. We do not restrict who the partners can be, provided that the criteria are met. Examples include a learner talking with another person who can be a peer, a teacher, a parent, or a computer agent…" (Chi and Wylie 2014a:223, italics in the original).
- *Constructive:* "Our taxonomy defines *constructive* behaviors as those in which learners *generate* or produce additional externalized outputs or products beyond what was provided in the learning materials" (Chi and Wylie 2014a:222, italics in the original).
- *Active:* "Learners' engagement with the instructional materials can be operationalized as *active* if some form of overt motoric action or physical manipulation is undertaken" (Chi and Wylie 2014a:221, italics in the original).
- *Passive:* The passive mode of engagement involves "learners *being oriented toward* and *receiving* information from the instructional materials without overtly doing anything else related to learning" (Chi and Wylie 2014a:221, italics in the original).

Suppose a learner, for example, was watching a video of an instructor demonstrating how to wash a window, as in the example described above. If the learner just watched the video without pausing it or doing anything else, then the activity would be classified as passive (i.e., paying attention). If the learner rewound the video to view portions of it over again or mimicked the actions of the teacher, then the activity would be classified as active (i.e., manipulating the given information without extending it). If the learner explained the actions by saying to itself, "The trajectory is intended to cover all the glass with minimal overlap," then the activity is classified as constructive (i.e., generating information not mentioned in the video). If the learner works with a second learner to co-construct some additional information, then the activity is classified as interactive (i.e., transactive collaboration). As an example, suppose one learner asks, "As long as we wipe every bit of glass, can we progress from bottom to top instead of left to right?" while another learner says, "Maybe that would drip dirty water on the clean glass." That is, they both construct information not presented in the video and build off each other's contribution.

This is exactly the kind of co-construction discussed earlier, except that the ICAP hypothesis allows two learners to be involved rather than a learner and a teacher. When ordered according to effectiveness for learning, the hypothesis ranks the four behavior modes as follows: interactive > constructive > active > passive. This is equivalent to: collaborate > generate > manipulate > pay attention.

According to the ICAP hypothesis, students learn the most when they collaborate and the least when they are merely paying attention. The ICAP hypothesis is consistent with a very large number of experiments. For example, Menekse et al. (2013) incorporated a topic in materials science as the target knowledge. They randomly assigned students to four groups, corresponding to the four ICAP modes. The students first took a test to determine how much they knew about the given topic prior to studying it (all four groups were about the same). Students then engaged in one of the following actions:

- They read a text passage (paying attention group).
- They read and highlighted important sentences within the text (manipulating group).
- They individually interpreted a graph that described the information contained within the text passage without access to the text (generating group).
- They interpreted the graph jointly with a peer without access to the text (collaborating group).

The groups then took a test to see how much they had learned. Test results were consistent with the ICAP hypothesis; that is, the collaborating group retained the most knowledge, followed by the generalizing, manipulating, and paying attention groups. The ICAP framework also hypothesizes cognitive processes that underlie the four modes and explains why they exhibit the observed ordering of effectiveness.

KLI Framework

The knowledge-learning-instruction (KLI) framework specifies three interacting taxonomies of kinds of knowledge, learning processes, and instructional methods (Koedinger et al. 2012). One fundamental underlying claim of this framework is that different instructional "treatments" are optimal for different kinds of "content" knowledge goals because of how those treatments best support the particular learning process relevant for that knowledge goal. In other words, "content-treatment interactions" are a common occurrence, whereby a particular instructional treatment (e.g., studying lots of examples) aids or accelerates learning for certain types of knowledge content (e.g., skills that implement multistep flexible procedures, such as math or science problem solving) but slows the learning of other types of knowledge (e.g., second-language vocabulary).

The bottom of Figure 11.3 depicts examples of three broad classes of knowledge: facts, rules (i.e., skills), and principles. On the left are three broad categories of learning processes: memory and fluency, induction and refinement, sense making and understanding. Also on the left are associated instructional treatments from "optimal scheduling," which enhances memory, through "worked examples," which enhance induction, to "accountable talk" (i.e., a collaborative dialogue prompting technique), which enhances sense making. The cells indicate cases in which the treatment (rows) was compared to a matched instructional control with regard to instruction of the indicated knowledge content (columns): +, 0, and – indicate a positive, null, or negative effect, respectively, of the treatment over the control on student learning outcomes. The bottom-left cell, for example, indicates that students learned Chinese vocabulary better when adaptive "optimal scheduling" selected their tasks, compared to receiving tasks in a fixed order.

The KLI framework provides empirical evidence for content-treatment interactions as well as a theoretical analysis to predict and explain when and why such interactions may occur. It is related to and largely consistent with the ICAP theory; however, one important difference is that ICAP does not consider as many types of knowledge as KLI, but instead focuses on more complex forms of knowledge. As a consequence, the claims made for the ICAP framework may only be relevant to "principles" in KLI framework terms. For

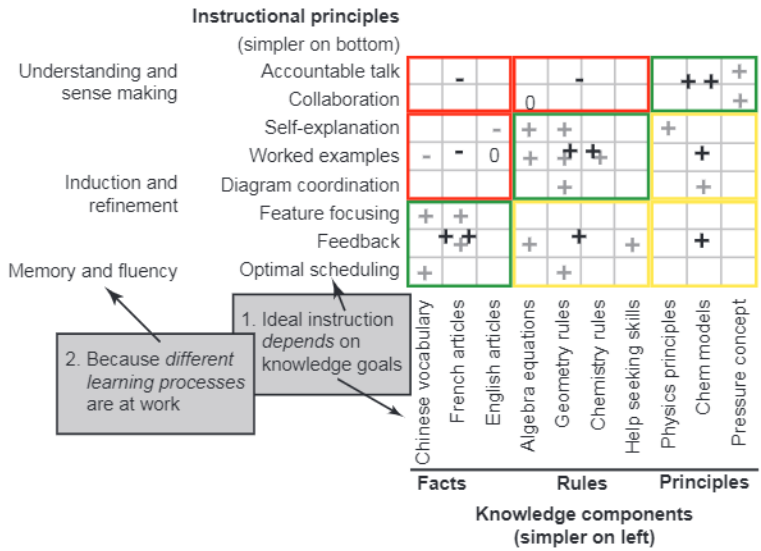

Figure 11.3 The KLI framework suggests "content-treatment interactions." Effectiveness of an instructional treatment (see "worked examples" row) depends on which learning processes it facilitates (e.g., induction) and inhibits (e.g., memory) and whether those learning processes are necessary (see "geometry rules" column) or not necessary (see "Chinese vocabulary" column) to meet the knowledge content goal.

example, according to KLI, forms of instruction that promote active rather than constructive learning may produce better outcomes when the goal involves less complex forms of knowledge, such as facts or skills/rules.

KLI and ICAP are directly relevant to ITL when a robot or software agent is the teacher and a human is the learner. Both frameworks indicate which instructional methods are most likely to work given the target knowledge content. KLI and ICAP may also be relevant when a robot or softbot is the learner, suggesting which instructional strategies and learning processes/mechanisms may be most effective for particular task domains.

Models for Task Instruction

Here we discuss the models that a teacher can hold with respect to an ITL learner, including models of the student, the domain/task, and a model of the pedagogical approach. We also discuss challenges related to communication with regard to models, as well as open modeling questions.

Overview of Models

The teacher in an ITL system (human or machine) must be able to assess the student's current learning state and make decisions that move the learner from one state to the next in a way shaped by the goal. This requires the teacher to maintain models of both the *student* and *domain*. These models may be created or evolve throughout a learning and co-construction process, wherein the teacher and learner develop an understanding of their shared world and/or each other. The teacher must also maintain and possibly evolve a model of a *pedagogical approach* in order to shape interactions.

The student model itself includes three components, the first of which is a model of the *learning state*, or the "knowledge overlay." This model incorporates a representation of correct knowledge that the student has acquired throughout the learning process and a representation of missing knowledge. The teacher maintains a model of each student with reference to the domain model, which can be very rich and may be represented as an "overlay" over the domain/task model. The second component is a model of the student's *misconceptions*, and the third is a model (or models) required for communication and interaction. This could include a model of modalities for *communication/interaction* (e.g., language, depiction) or abstractions over the level of concepts to communicate.

The teacher also holds a model of the *domain/task* that is separate from the student model. The domain model is maintained in the teacher's mind and is representative of what experts know as well as what intermediate and/or novice students know. It can also include conceptual alternatives of the task. The domain model can change in structure or content over time.

Learning State Model

The model of the learning state is analogous to the tutoring system "student model," which refers to information that represents what the student currently knows and has accomplished so far. This includes the history of interaction over time and is not simply a snapshot of the current state. The process by which the student arrives at the current state is important and holds a lot of information for the teacher, who is shaping the interactive experience toward some learning.

This information serves one or more of the following purposes:

- It assigns or recommends a task for the student to perform next.
- It reports the student's current state to the instructor.
- It reports the student's current state to the student.
- It guides the system's selection of feedback, hints, or other scaffolding.

Sometimes the student model contains only simple information. For instance, if the tutoring system's only use for the student model is to meter the progression through a linear sequence of tasks, then the student model need only record where the student is located in the task sequence. If the system's only use for the student model is to decide when the student has correctly accomplished three of the previous four tasks, then it only needs to track how many of the last four tasks have been completed.

However, many systems incorporate student models that are based on assessing what knowledge students have learned so far, rather than just which tasks they have completed. Such systems often divide to-be-learned knowledge into pieces, typically referred to as "knowledge components" (if they are relatively small) or "instructional objectives" (if they are larger). If a tutoring system incorporates knowledge components, it often represents which ones a given student knows using a simple data structure called an "overlay." In its simplest form, an overlay is a single binary variable (mastered/unmastered) per knowledge component; another common representation is a value between 0 (completely unmastered) and 1 (completely mastered). It is traditional (albeit confusing) to refer to a student model that uses only correct knowledge components as an "overlay" model, whereas a student model that includes both correct and incorrect knowledge components is referred to as a "buggy" student model.

Model of Misconceptions

The tutoring agent in an ITL system may also maintain a model of student misconceptions. This model can contain the misconceptions held by multiple students or those of a particular student. If the tutor agent understood the learner's misconceptions, the tutor could potentially be able to exploit this knowledge to enhance the effectiveness of its teaching; this has formerly been studied as

the "diagnostic remediation hypothesis." Evidence, however, rejects this hypothesis, at least under certain conditions. Study results from Putnam (1987) and Siler and VanLehn (2015) have shown that human teachers rarely utilize knowledge about students' misconceptions, even when this information is explicitly presented to them. Other studies provide evidence that use of knowledge about misconceptions does not help students to learn algebra (Sleeman et al. 1989). This may be due to the fact that misconceptions rarely result in systematic errors. Researchers have also found that the explicit remediation of misconceptions (or "mind bugs") can have a positive effect when learning about physics (Albacete and VanLehn 2000).

Although evidence of the usefulness of knowledge about misconceptions is mixed, and greatly depends on how systematic the errors are, it should be noted that human teachers maintain several conceptions about how a learner works. This is crucial for adapting language and level of explanation to a target audience (e.g., children of different ages or adults with different levels of expertise).

Task Domain Models

In the expert, or domain, model of ITL, the system stores its knowledge relevant to solving tasks within the given domain. In a subclass of intelligent tutors, sometimes called "cognitive tutors," there is a commitment that the domain model is a "cognitive model," one that contains knowledge able to produce a broad range of possible solutions for domain tasks, including variations in expert solutions as well as in student solution strategies, both correct and incorrect. This cognitive model may also include knowledge about a progression (or "learning trajectory") of approximate conceptions or misconceptions. For example, in human biology, the model may include a misconception that the human circulatory system consists of a single loop involving only the heart, or of one loop involving both the heart and lungs, in addition to the correct conception that there are two loops: one involving the heart and one involving the lungs.

This domain model can provide adaptive tutoring to students at multiple temporal grain sizes of interaction and adaptivity. Using a model-tracing algorithm, the tutor can monitor student task performance in comparison with the task performance the domain model can generate (i.e., a kind of plan recognition in AI terms). Steps taken during a student's performance of the task are correct when they match application of correct knowledge in the domain model and are incorrect when they match application of incorrect knowledge (in which case specific feedback can be provided to explain why the given step is incorrect) or when there is no match (in which case a simple error feedback indication can be provided). When students have difficulty completing a task, they can ask for an instructional explanation or demonstration of the next step,

which would provide the means for students to learn how to perform these steps on their own in the future.

Model tracing of each student step yields an "overlay" on the domain model, indicating the probability that each knowledge component is known for each student. We refer to this above as the model of the student learning state. A knowledge-tracing algorithm updates these probabilities as students perform correct or incorrect steps during task performance, and the results are used to adapt future task selection to facilitate cognitive mastering of the correct knowledge components. For cognitive tutors, the tutoring interaction is dependent on the completeness and quality of the domain (or "cognitive") model. Thus, the broadest loop of adaptivity is to use student performance data to identify flaws in the domain model (e.g., common learning challenges that are not independently represented) and to improve it and the associated task design (Aleven et al. 2016).

Pedagogical Approach/Expertise Model

One of the goals of ITL systems is to design a robotic agent capable of learning a task from a human tutor, performing the task, and potentially to teach the task to a different human learner. Task knowledge must, therefore, be combined with didactic/pedagogical knowledge. To realize this goal, one possibility is that the agent reuses the teaching strategies incorporated when the agent itself first learned the task. Alternatively, a didactic/pedagogical model could be included in the agent; the agent would then apply this model to the task knowledge to generate instructions or other tutoring aids.

Model-Based Scheduling

Thus far, the intelligent tutoring system research community has focused primarily on curriculum adaptation (i.e., which content or problem to present next) and support adaptation (particularly with regard to changing the type of feedback provided) based on a student model. These adaptations address the "what" and the "how" of instruction; however, there is a new, promising area of research and development that focuses on the "when" of instruction (Gluck et al. 2019). Here, the emphasis is on personalizing the scheduling of learning events based on performance history. The adaptation is conducted at the level of individual knowledge components, a collection of which would comprise proficiency in problem-solving or task completion.

Personalized scheduling of learning depends on the availability of a computational model of the dynamics of human learning systems (Raaijmakers 2003; Pavlik and Anderson 2005; Walsh et al. 2018). The general idea is to calibrate parameters based on accumulated empirical evidence, then predict forward in time to determine a good schedule for the next study or practice

opportunity. What represents a "good" schedule depends critically on whether the bias is toward rapid initial acquisition or longer retention of information. This trade-off relates to a phenomenon known as the "spacing effect" (Cepeda et al. 2006; Benjamin and Tullis 2010). If rapid acquisition is more important, then cramming learning events into a tightly massed sequence is preferable. This will generally accelerate the initial acquisition of that material but will also reduce the duration of retention. If better retention is more important, then spacing repeated learning events over a longer period of time is preferable, up to a point. If temporal spacing is too long early on in the process, then people forget the information more quickly than they learn it, and the potential advantage of distributing the practice over time is lost.

The implication for ITL is that these predictive, adaptive personalization capabilities can be implemented in the machine intelligence and applied in any context in which one of the agents participating in the interaction is a computer.

Communication with Respect to Models and Asymmetries between Human and Machine Learners

One key to a successful interaction and, by extension, to a successful tutoring interaction is that the teacher and learner share sufficiently similar communication processes and concepts; this is often referred to as "common ground" (Clark and Brennan 1991). When the teacher is tutoring, an implicit or explicit assumption is made by the teacher that the learner can correctly interpret the teacher's communicative signals. This requires a shared set of communicative signals to be interpreted in a manner considered successful by the teacher (or another external observer).

Systems for tutoring humans are designed based on assumptions of the students' knowledge base (e.g., what the learner knows, what the learner does not know, common ground, language, how the learner learns). When designing a machine to teach a human, we leverage substantial knowledge about how humans communicate and learn. However, when the learner is a machine, the validity of many of these assumptions is weakened or disappears. When a human teaches a machine, it is more difficult to make *a priori* assumptions about the types of communication available. Systems capable of constructing and building upon a model of language are crucial. Depending on the complexity of the learned task, a significant effort may be required to design communicative modalities that allow instructions and knowledge to be transferred from a human teacher to an artificial learner. Forbus et al. (2017) demonstrated a system that integrates visual processing, spatial representations, and conceptual knowledge; for the system to learn successfully from a human tutor, it needs to establish a common set of communicative signals (language and visual sketching, in this case) and requires access to the ontology of the world.

Similarly, when a machine teaches a human, the machine requires a model of the human learner, but it is challenging to encode all the relevant

information. Asymmetries also exist between the representation and use of the learned knowledge. A human may learn to perform a task and use this as a basis to teach the task to another learner. A machine may be able to teach based on its learning mechanism, or teaching must be addressed separately.

Human–robot interaction exaggerates the problems associated with asymmetry, as in the correspondence problem. First identified in the context of imitation learning (see, e.g., Nehaniv and Dautenhahn 2007), the correspondence problem relates to the difficulty of mapping actions from one body to another. If a human demonstrates a skill to an agent with a dissimilar body plan or dissimilar actuation (e.g., a robot arm with five degrees of freedom that is learning to pour a drink by observing human demonstrations), how is the learning agent to map this demonstration to its own capabilities? The correspondence problem also holds in ITL: while the interactive element allows skill transfer to be scaffolded more gradually, the correspondence problem still requires a solution.

Methods for Task Instruction

There are myriad methods of instruction that are natural to human–human interaction, but whether emulation of these methods can or will support ITL is an open question. These instruction methods include *signal-focused methods*, which involve the modulation of signals such as speech and gesture, and *directing/attending behaviors*, such as pointing as well as other methods such as coupling of language with action. All instruction methods involve a persistent process of *monitoring*, *adjustment*, and *feedback* based on the projected cognitive process of the given student. This persistent process is important for the teacher's choice and adjustment of feedback type, including timing considerations.

Signal-Focused Methods

Developmental studies have addressed some of the scaffolding methods that people use to improve understanding of action and speech: motherese, motionese, gesturese. Collectively known as the "eses," these methods may be of fundamental interest to ITL. As techniques used to make instruction more effective and efficient with children, they may generalize to make instruction more effective and efficient with machines:

- *Motherese*, parentese, or child-directed speech refers to modifications in verbal behavior (Dominey and Dodane 2004). These modifications can be performed on all linguistic levels, including prosody (speaking with higher pitch and long pauses), semantics (making references to

here and now, regular use of repetition, lower levels of abstraction), and grammar (shorter, simpler sentences).

- *Motionese* concerns the performance of actions. As reported by Rohlfing et al. (2006), child-directed actions are performed with rounder (smoother) movements and more frequent pauses.
- *Gesturese* refers to how caregivers modify the frequency, type, and duration of their gestures when they talk to children. Grimminger et al. (2010) reported that gesture frequency increases when teachers issue instructions for difficult tasks, or when they instruct children at risk of delayed language development (compared to typically developed children).

Learner feedback is crucial in these signal-focused methods and must be kept in mind when applying them to interactions with a machine. As demonstrated by Lohan et al. (2012), it is important for a robot to behave in a contingent manner and react in a timely manner to the communications initiated by the tutor. Fischer et al. (2011) indicate how tightly the "eses" are tailored to the learner: if the robot reacts to the tutor's instructions using visual behavior alone (e.g., eye gaze), then the tutors will modify their behavior in this modality. Consequently, behavior modifications can be observed in the form of "motionese" (visually perceptible), not in "motherese," as this would most likely require a verbal response from the robot. This research suggests that if a robot indicates sensitivities toward interaction in the form of a contingent behavior and particular modalities of communication, then tutors can provide beneficial input in the form of parentese, motionese, and/or gesturese.

Showing, Pointing, and Depicting

When children are already experienced with regard to interaction, their learning can be guided through "social cues" embedded in a sequence of particular actions. For instance, a pointing gesture is embedded as a social cue within a referential frame (e.g., eye gaze, pointing to an object, labeling it). Such sequences (pointing, showing, depicting) engage children to guide their attention and actions toward a specific goal.

When showing an object or performing a task, language and action often are synchronized such that they reinforce each other. This intermodal redundancy foregrounds key information (Bahrick et al. 2004) and creates meaningful "acoustic packages" (Hirsh-Pasek and Golinkoff 1996) to facilitate comprehension and learning. It is a practice that is generally useful in communication and is especially helpful with young children. Among infants who cannot yet understand semantics, labels or words were found to highlight the commonalities between objects and situations, facilitate object categorization, and override the perceptual categories of objects (Rohlfing and Tani 2011). Conversely, when action is provided concurrently with speech,

it seems to embody the meaning of language, even in young infants (e.g., Nomikou et al. 2017).

Directing, Performance Modeling, and Peer Performance

Musical instrument instruction is an interesting practice to consider in the context of ITL, as it involves humans executing physically demanding tasks in the real world, with multidimensional success criteria. The ontology of feedback on music instrument instruction, provided by Yee-King et al. (2014), indicates the diversity of success measures in music instrument learning (e.g., timbre, groove, and articulation). Musical instrument tutors employ three key techniques that can be applied in ITL scenarios:

1. *Directing* involves the student playing a piece of music, with the tutor providing verbal and gestural directions during the performance, similar to a conductor. For example, the tutor might say "build up the speed now," or "careful with the fingering for the right hand." The information is highly contextual and can be directly acted upon by the student to improve performance. This approach could apply to an ITL scenario in which a machine is learning to set a table: "careful with the placement of the fork, as there's no space for the plate."
2. *Performance modeling* involves the tutor playing the instrument and presenting the student with both good and poor examples of how to play sections of a given piece of music. In the context of ITL, this correlates to a tutor providing a set of carefully chosen, labeled examples.
3. *Peer performance* involves students performing a complete piece for an audience of their peers as well as the tutor. At the end of the performance, the peers are encouraged to give feedback to the performers. Peer performance is important for the performer, as it takes place in a context where failure has a high cost (embarrassment) and closely mirrors the actual context in which music is commonly performed. It is also important for the listeners, as they learn to discern between good and poor execution. Both performer and listeners gain a wider view of the performance: What effect did it have on an audience? How does the sound change when performing in a concert hall? How does a person's performance respond to the pressure of being observed? This raises the intriguing idea of one robot washing a window while others critique its performance.

Monitoring, Adjustment, and Feedback

Feedback is especially important, perhaps essential, to almost all instructional interactions. Feedback can be provided/received at any time during

a task: at the beginning to inform upcoming action on the basis of prior performance, in real time during task performance, and at the end when the task is complete. Many details need to be considered when providing and receiving feedback (e.g., timing, synchrony of cues, task structure, conventional social norms).

Feedback received by a teacher or learner can trigger a process whereby the agents interact to repair incorrect inferences or clarify ambiguities. Feedback received by the learner can also trigger a feed-forward process, whereby the learner is able to apply feedback received from a teacher to future attempts at completing a task, in the absence of the teacher's guidance (Hattie and Timperley 2007). The feed-forward process maps across several desirable aspects of ITL as follows:

- It results in an efficient use of instructor time, since feedback is reused.
- It signifies a transition on the part of the student from being unconsciously incompetent to consciously incompetent: students can detect their own mistakes and use previously received feedback to rectify them.
- The student switches his/her role to that of his/her own instructor.
- It includes metacognitive aspects, wherein the student becomes aware of his/her own learning process.

During human–human interactions for learning and teaching, particularly between parents and infants, monitoring the learner is important. Teachers modify their behavior online in a moment-to-moment co-construction toward a goal that is shared with the learner, with interactional loops between teacher and learner, such as between the teacher's hand movements and the learner's eye gaze (Pitsch et al. 2014). Study results have shown that these processes translate to human–robot interactions, with the robot as the learner and the human as the teacher, under certain conditions (Vollmer et al. 2009; Pitsch et al. 2013). These conditions include a certain appearance and social behavior (such as feedback or contingency) on the part of the learner that forms a social interaction interface for the human to employ during teaching. It has been shown that the way in which an action demonstration is instantiated and how the action is structured (in terms of highlighting what about it is important when teaching the action to a robot) strongly depend on the learner's feedback (Vollmer et al. 2014). In a case involving a robot learner, these interactional loops with the adaptive behavior of the teacher could be exploited for the benefit of learning. The robot could, through its feedback, elicit certain adaptations to the teacher's behavior, allowing for testing of a hypothesis about, for instance, the importance of a certain part of a demonstration.

The lessons gained through human–human learning/teaching interactions are most clearly relevant to robot learning, although some of these teaching methods have analogous forms in the virtual world of graphical user interfaces

within which software agents learn. The signal-focused and directing and attending methods discussed above may not be implemented in the same way in a graphical user interface, but similar implicit communicative functions can be achieved in other ways. Analogous to a robot finger pointing at (or an eye gaze directing attention toward) an important element or feature within an object or scene, features may be highlighted through other means, such as bolding, circling, or underlining. Such "feature focusing" has been used to enhance human learning; for instance, to help teach Chinese symbols by highlighting semantically meaningful components (known as "radicals") within them. It has also been used to aid machine learning of grammatical structure in algebra (Li et al. 2015) by highlighting elements that interrelate (e.g., highlighting "–3" in "–3x"). Do other aspects of human–human learning/teaching interactions have relevance within the virtual world of agent learning? This is an interesting question that awaits further exploration.

Learning by Teaching

When the primary goal is for a (human) student to learn a task, studies have demonstrated that it is often more effective to assign a human the role of a teacher, who then must tutor a computer agent by demonstrating, for example, how to solve algebra equations while providing feedback about the attempts (Chase et al. 2009; Matsuda et al. 2013). In these "teachable agent" applications, the computer agent takes a quiz after being instructed by the human, and the agent's performance provides students with feedback on the ways in which they succeeded or failed to teach the agent. This method of learning proves to be an effective "ego-protective buffer" whereby students' knowledge gaps are revealed by their agent's performance rather than their own.

When the primary goal is for an artificial agent to learn a task, it may also be valuable to swap human and computer roles so that the agent becomes the teacher and the user becomes the learner. One such scenario is the use of the ITL agent to help teach or "entrain" humans to use the language understood by the ITL agent. Consider a situation in which the human says, "Start up my mail" to the agent, rather than "Open the email app." In response, the agent might say, "Well, show me," and the user subsequently succeeds in teaching the agent via demonstration. Key to this example, the learning agent then switches into the teaching role and explains the procedure it learned back to the user, using terms the agent understands (e.g., "First, I opened the email app. Then, …"). Upon observing this explanation, the user may have a better chance in subsequent attempts of using language patterns that the agent understands. In addition, the agent has had the opportunity to expand its vocabulary.

Closing Thoughts

Structural and Adaptive Perspectives on Improving Instruction

Instructional improvements based on education technology (including ITL) can be broadly classified as adaptive, structural, or both adaptive and structural (VanLehn 2016). Instructional improvement is adaptive if the content and structure of the new instruction are the same as in the baseline instruction, but the new instruction interacts with learners differently based on their performance. An instruction improvement is considered structural if the structure, or plan, of the instruction differs significantly from the baseline instruction. Consider the following examples:

- Purely structural: A new, three-week module in a middle-school science class is revised to focus on accessing data obtained from a real radio telescope.
- Purely adaptive: An organic chemistry practice system contains the same large set of problems as before, but the system now monitors the students' successes and failures, and recommends the best problem to address next.
- Both structural and adaptive: A collaborative learning system has students working in small groups to build an economic model. The system monitors their interactions and sends advice covertly to group members who are not speaking up enough or who are dominating the group dynamics.

Regardless of whether one is designing instruction that is adaptive, structural, or both, there are several choices to consider, including demonstrating, telling (through verbal instruction), and providing feedback (e.g., as accomplished through depicting, showing, pointing to, or coupling language and action). Instructional strategies include numerous choices with regard to structuring the environment, choosing learning examples, comparing or contrasting tasks or problems, making sequencing or curriculum decisions, and determining how elaborately to expose the steps and whether, when, and how to provide reward or feedback (Koedinger et al. 2013).

Historical Perspective on Programming and Implications for ITL

In the early days of programming, FORTRAN, COBAL, and LISP were the only languages available. Many people thought that redesigning the languages would make programming significantly easier. This led to ALGOL, PASCAL,

and eventually myriad languages currently in place, which may be a little easier to use than the first languages, but certainly did not vastly simplify programming. Similarly, natural language programming, graphical programming languages, and programming-by-demonstration did not have significant impacts, as they proved useful only within a limited range of programming problems.

Perhaps the biggest advance in software engineering has been the recognition and systematic development of agile programming methods (Beck et al. 2001). Prior to this recognition, the orthodox method of programming (now called "waterfall") involved three phases:

1. Writing detailed specifications in natural language, and sometimes diagrams (e.g., flow charts; UML).
2. Converting the specification into a giant program.
3. Testing the program.

In contrast, the agile method divides the overall function of the desired program into many small pieces, and follows the following three-step approach to develop each piece:

1. Write a specification of the new piece of functionality, called a user story.
2. Add that function to the program.
3. Test the new function while also testing that the preexisting functions remain intact.

Nowadays, the waterfall method is typically used when developing a program that has been previously developed but needs to be repurposed for a new situation. Under these circumstances, the complete functionality of the program is well known, so the specifications can be accurate and complete. Most other programming is performed using agile programming, because it allows the ultimate users of the program (often called "clients") to use the emerging program as functions become available, and to provide new user stories or revisions to old user stories. In other words, agile programming is more interactive.

The ITL vision is similar to these historical features of programming methods in many respects. First, it is interactive: like agile programming, the interactivity of ITL is probably the most powerful simplification of the overall job of creating engineered activity. Second, ITL combines natural language and demonstration as ways of expressing user stories. A skeptic may note that natural language and demonstration provided only limited simplification of traditional programming and that perhaps an implication of this is that they may have limited importance for ITL, as well. However, it is important to keep in mind that the traditional programming model of machine as a tool (rather than a partner) and implementation for niche specialization (rather than

generalized interactive learning of new tasks) inherently limited the potential utility of natural language and demonstration.

One way to understand ITL's potential is to consider its scope. It is clearly not intended to replace traditional programming for large systems; ITL will most likely replace scripting, writing simple programs of perhaps a page or two. More specifically, it may fit best in the middle of a continuum of ways that users can express their intentions to computers. On the simple end of this spectrum are menus that allow a user to choose from a small set of predefined behaviors. Next, in terms of simplicity, are forms that have a set of menus, type-in boxes, buttons, and other controls to allow users to express somewhat more complicated intentions. To indicate intended behaviors that go beyond those readily designated by form-based user interfaces, we currently write scripts, demonstrate the intended behavior to a macro-writer, or both. This capability is likely to be replaced by ITL. The next step up in terms of complexity will likely require traditional programming.

The same complexity continuum applies even when all agents are human, such as when an employer describes intended behaviors to employees. On the simple end, a short phrase suffices to explain the boss's intentions (e.g., "Hold my calls.") On the more complex end, modern organizations use formal languages, often called *business process languages*, or procedure manuals. An ITL system would most likely incorporate intended behaviors "taught" to human workers through a short email message or narrated demonstration.

Based on these analogies, it seems likely that ITL systems may address a "sweet spot" within the continuum of complexity, resulting in many potential applications. Moreover, these applications are poorly served today because the scripting/macro-demonstration process is manually intensive.

Similarities and Differences between ITL and Human/Animal Teaching and Learning

To clarify similarities and differences between machine and human/animal learning, consider the following examples. The first involves a "cat flap" (i.e., a portal that allows a cat to enter or exit a house at will): through a system of rewards (and some patience), most cats learn to open and pass through the cat flap. This is an example of operant conditioning. The second involves standard computer programming: a computer is initially unable to perform a given task (e.g., play chess), so the user provides it with a set of instructions. Initially, the instructions are of poor quality, and the computer is still unable to perform the task. Responding to this failure, the user continually updates the instructions until the computer succeeds. This scenario is both interactive and involves the learning of a task.

Note the cat flap example does not involve intelligent behavior or reasoning; it is a primitive form of teaching (operant conditioning). Once learned, however, the cat would be quite flexible with regard to its new task; that is,

confronted with a new cat flap or the same cat flap in another door in the same house, the cat would most likely succeed in going through the flap. Also, the cat can learn many other tasks through operant conditioning without interfering with its ability to maneuver through cat flaps. In contrast, the computer programming example involves hardly any learning by the computer, whereas the task for the teacher is immense.

Humans and animals have many advanced, fine-tuned, and mutually aligned learning mechanisms. Human–human ITL begins with learning through reinforcement in the first months of life, followed by demonstration and imitation, and later on through exploration, instruction, and reasoning. It is quite clear that our current AI systems and robots lag far behind in this respect: they may be very good at applying a single learning mechanism (e.g., learning through examples or parameter optimization), but the broad integration and application of multiple learning mechanisms in complex, real-time task learning and performance remains elusive. We aspire to create artificial systems that combine diverse mechanisms for co-constructive task learning and instruction.