Fehlererkennung und Fehlerdiagnose für verläßliche Systeme Automatisierungstechnik vs. verteilte Systeme[1]

Felix C. Gärtner
Fachbereich Informatik
Technische Universität Darmstadt
felix@informatik.tu-darmstadt.de

Armin Wolfram
Institut für Automatisierungstechnik
Fachbereich Elektrotechnik und Informationstechnik
Technische Universität Darmstadt
AWolfram@iat.tu-darmstadt.de

Fachbereich Informatik
Technische Universität Darmstadt
Technischer Bericht TUD-BS-1999-03

6. Juli 1999

[1]Diese Arbeit entstand im Rahmen des DFG-Graduiertenkollegs "Intelligente Systeme für die Informations- und Automatisierungstechnik" an der Technischen Universität Darmstadt.

1 Einführung

Im Zuge wachsender Automatisierung und Rationalisierung und den damit verbundenen gestiegenen Anforderungen an *Zuverlässigkeit*, *Lebensdauer* sowie *Funktionsgarantien* technischer Systeme hat das Interesse in Wissenschaft und Industrie an einer umfassenden Überwachung und Fehlerdiagnose technischer Anlagen stetig zugenommen. Die Aufgaben der Überwachung bestehen darin, den gegenwärtigen Systemzustand anzuzeigen, unerwünschte oder unerlaubte Systemzustände zu melden und im Fehlerfall geeignete Maßnahmen einzuleiten, um den weiteren Betrieb der Anlage sicherzustellen bzw. Schäden und Unfälle zu vermeiden. Bei der modernen Fehlerdiagnose ist neben der bloßen Fehlererkennung auch eine Bestimmung des Typs, der Größe und des Fehlerorts vorgesehen. In der Wissenschaft hat jedoch die Komplexität technischer Systeme in verschiedenen Technologiebereichen zu einer starken Ausdifferenzierung und einer verwirrenden Vielfalt im Hinblick auf Terminologie und Methodik geführt. Bereiche, die an sich ähnliche Themen behandeln, erscheinen für auf dem jeweils anderen Gebiet Forschende fremd und verwirrend. Dabei steckt in einer stärkeren interdisziplinären Betrachtung mancher Sachverhalte viel Potential für das Verständnis und die Weiterentwicklung der zugrundeliegenden Themen.

Wir wollen dieses Diktum überprüfen anhand eines einheitlichen Themengebietes und zweier Forschungsbereiche, die normalerweise ein getrenntes Leben führen. Das Themengebiet lautet "Fehlererkennung und Fehlerdiagnose" (wobei wir den Schwerpunkt auf die Fehlererkennung legen) und die Forschungsdisziplinen sind einerseits die Automatisierungstechnik (normalerweise ein Teilgebiet der Elektrotechnik) und andererseits das Gebiet der Verteilten Systeme (üblicherweise ein Teilgebiet der Informatik). Wir stellen die in den jeweiligen Disziplinen üblichen Verständnisse des Themengebietes vor und diskutieren schließlich Forschungsrichtungen, in denen bereits Überschneidungen von Terminologie und Methodik existieren, bzw. solche, bei denen noch Potential für einen Ausbau der Zusammenarbeit bestehen. Da sich dieser Beitrag an ein informatisches Forum wendet, werden die Belange tendenziell aus der Sicht der Informatik dargestellt.

2 Fehlererkennung und Fehlerdiagnose in der Automatisierungstechnik

Systemannahmen, Fehlermodellierung. Unter einem *technischen System* (auch *Prozeß* oder *Strecke*) versteht man eine abgegrenzte Anordnung, in der eine Umformung und/oder ein Transport von Materie, Energie und/oder Information stattfindet [17]. Die Prozeßgrössen sind dabei im allgemeinen kontinuierliche Signale, die in zeitlich beschränkten Perioden abgetastet werden. Fehler in derartigen Systemen lassen sich auf zwei unterschiedliche Arten modellieren. Einerseits spricht man von *additiven Fehlern*, wenn den Prozeßgrößen ein zusätzlicher Offset überlagert ist, während *multiplikative Fehler* ihre Ursache zumeist in geänderten Zeitkonstanten oder Verstärkungsfaktoren haben [15, 20].

Methoden der Fehlererkennung. Zur Realisierung von Fehlererkennung und -diagnose an technischen Prozessen bedient man sich charakteristischer Prozeßmerkmale, die einerseits auf *analytischem Wege* gewonnen werden, zum anderen aber auch auf *heuristischem Prozeßwissen* beruhen (vgl. Abb. 1). Bei der Generierung *heuristischer Symptome* berücksichtigt man qualitative Informationen, über die im wesentlichen menschliche Bediener verfügen. Die Palette reicht dabei von Geräuschen, Farben, Gerüchen, Schwingungen bis hin zu Verschleißmarken. Darüber hinaus berücksichtigt man zusätzliche Informationen, welche die bisherige Prozeßgeschichte in Form von Wartung, Reparatur, usw. einbeziehen. Die mathematische Beschreibung heuristischer Symptome erfolgt häufig unter Verwendung linguistischer Variablen (z.B. "klein", "mittel", "groß") mittels Fuzzy-Logik [13, 19].

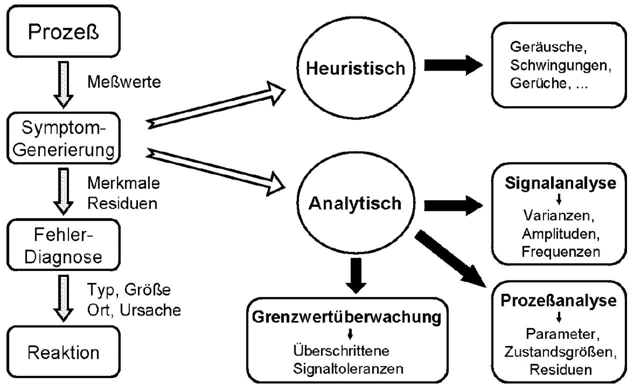

Abbildung 1: Schema zur Generierung heuristischer und analytischer Symptome.

Die einfachste Variante zur Erzeugung *analytischer Symptome* besteht darin, meßbare Prozeßgrößen (z.B. elektrische Ströme), auf das Überschreiten von Schwellwerten hin zu überprüfen (sog. Grenzwertüberwachung). Neben der Festlegung geeigneter Schwellwerte besteht die Hauptschwierigkeit dieser Methode darin, daß Alarmmeldungen erst nach großen Merkmalsänderungen erfolgen und eine detaillierte Fehlerdiagnose im allgemeinen nicht möglich ist [20]. Leistungsfähiger sind dagegen *signalbasierte Methoden* (sog. Signalanalyse), bei denen die zeitlichen, spektralen oder auch stochastischen Eigenschaften von Meßgrößen des technischen Prozesses überwacht und auf charakteristische Merkmale hin untersucht werden [18]. Bei Lagerschäden in mechanischen Bauteilen treten beispielsweise spezielle Anteile im Frequenzspektrum der Drehzahl auf, die mit den bekannten Methoden der digitalen Signalverarbeitung extrahiert werden können [22, 27].

Eine Fülle weiterer Methoden zur Fehlererkennung beruhen auf *analytisch* oder *experimentell* gewonnenen *Prozeßmodellen* (sog. Prozeßanalyse, man nennt diese Verfahren oft auch *modellbasiert*). Die dynamischen bzw. statischen Modelle liegen dabei in Form von Differentialgleichungen, Differenzengleichungen oder Kennfeldern vor und geben auf diese Weise die Abhängigkeiten der verschiedenen Ein- und Ausgangssignale des Prozesses wieder [15, 17, 24]. Das Systemverhalten dynamischer, nichtlinearer Prozesse läßt sich in allgemeiner Form durch folgendes Gleichungssystem beschreiben, wobei $\underline{u}$ und $\underline{y}$ die Ein- bzw. Ausgangsgrößen, $\underline{x}$ die internen Prozeßzustände und $\underline{\theta}$ die Parameter des Prozesses symbolisieren (unterstrichene Variablen bezeichnen Vektoren).

$$\frac{d\underline{x}}{dt} = \underline{f}(\underline{x}, \underline{u}, \underline{\theta}) \qquad (1)$$

$$\underline{y} = \underline{g}(\underline{x}, \underline{u}, \underline{\theta}) \qquad (2)$$

Eine Möglichkeit der Fehlererkennung besteht nun darin, die Parameter $\underline{\theta}$ im laufenden Prozeß zu schätzen und damit Änderungen zu erkennen [16–18]. In einem weiteren modellgestützten Verfahren werden aus den Modellgleichungen 1 und 2 sogenannte *Paritätsgleichungen* generiert, die nur von den bekannten Ein- bzw. Ausgangsgrößen $\underline{u}$, $\underline{y}$ des Prozesses abhängen und damit unabhängig von den unbekannten internen Prozeßzuständen $\underline{x}$ sind.

$$\underline{r} = \underline{h}(\underline{u}, \underline{y}, \underline{\theta}) \qquad (3)$$

Die Funktion $\underline{h}$ enthält gewissermaßen das Modell, gegen welches das System verglichen wird. Auf diese Weise lassen sich sog. Residuen $\underline{r}$ berechnen, die nur im Fehlerfall merklich von Null abweichen und dadurch einen Fehler anzeigen [15, 16, 18].

Methoden der Fehlerdiagnose. Das Problem der Fehlerdiagnose stellt ausgehend von den gewonnenen Symptomen im wesentlichen ein Mustererkennungsproblem dar, das mit bekannten *Klassifikationsverfahren*, oder *Neuronalen Netzen* gelöst werden kann [5, 25].

3 Fehlererkennung und Fehlerdiagnose in verteilten Systemen

Systemannahmen und Fehlermodellierung. Bei den Systemmodellen unterscheidet man in den verteilten Systemen grundsätzlich zwischen *synchronen* und *asynchronen* Systemen. In asynchronen Systemen existieren keinerlei Annahmen über maximale Nachrichtenlaufzeiten oder den maximalen Gangunterschied zweier prozeßlokaler Uhren [26]. Oft wird beispielsweise das Internet als asynchrones System bezeichnet, da die geographische Größe und keinerlei Kenntnisse über die Last auf einzelnen Vermittlungsknoten zu nahezu beliebigen Zeitverzögerungen führen können. Generell ist es beim Systementwurf auch einfach nur "praktisch", ein asynchrones System vorauszusetzen, da es oft schwierig ist, durch ein Protokoll vorausgesetzte Zeitannahmen im unterliegenden Nachrichtensystem zu garantieren; wenn ein Protokoll im asynchronen System funktioniert, dann funktioniert es auch in allen anderen Systemen.

Fehler werden in verteilten Systemen durch gezielte Veränderung von Programmcode oder (auf höherer Abstraktionsstufe) durch eine Abschwächung von Annahmen an der Kommunikationsschnittstelle eines Rechners modelliert [9]. So versucht man, die vorübergehenden oder permanenten Zustandsveränderungen nachzubilden, welche Fehler nach sich ziehen. Vor jeder Zuverlässigkeitsbewertung müssen also zunächst konkrete Fehlerszenarien ausgewählt werden, gegen die ein Fehlertoleranzverfahren getestet werden soll [11]. Dies ist die sogenannte *Fehlerannahme* [10]. In den verteilten Systemen haben sich einige Standard-Fehlerannahmen etabliert, wie z.B. "Absturz" (engl. *crash* oder *failstop*), "Verletzung von Echtzeitschranken" (engl. *performance*) [21], "vorübergehende Störeinflüsse" (*transient* [1, 4]) oder beliebiges, sog. "byzantinisches" Fehlverhalten [23]. Bei *Absturz* geht man beispielsweise davon aus, daß ein Rechnerknoten plötzlich jegliche Kommunikation mit seinen Nachbarn dauerhaft abbricht. Man nimmt jedoch zusätzlich an, daß nur eine beschränkte Zahl von Knoten gleichzeitig dieses Verhalten aufweisen.

Methoden der Fehlererkennung. Neueste Forschungsergebnisse [2, 14] haben formal nachgewiesen, daß jedes Fehlertoleranzverfahren mittels zweier simpler Komponenten konstruiert werden kann. Diese Komponenten werden *Detektoren* (engl. *detectors*) und *Korrektoren* (engl. *correctors*) genannt (sie sind im wesentlichen Formalisierungen der Eigenschaften von Sensoren und Aktuatoren). Detektoren sind Einheiten der Fehlererkennung (engl. *failure/fault detection*), Korrektoren Einheiten der Fehlerbehebung (engl. *recovery*). Die Fehlererkennung bezieht sich in der Regel auf die zugrundeliegende Fehlerannahme, d.h. ein Fehlerdetektor unter der *crash*-Fehlerannahme meldet, ob ein Rechnerknoten abgestürzt ist oder nicht, oder ein Fehlerdetektor unter der *transient*-Fehlerannahme meldet Inkonsistenzen im lokalen Zustand eines Prozesses [1].

Generell besteht das Problem der Fehlererkennung aus der Entdeckung eines Prädikates P auf dem globalen Systemzustand [14], was wegen der fehlenden globalen Sicht in großen verteilten Systemen nur unter hohem Aufwand erreicht werden kann [7, 8, 28]. Oft genügt es jedoch, lokale Inkonsistenzen durch die Einbringung von Redundanz zu erkennen (z.B. durch fehlererkennende Codes bei der Nachrichtenübertragung).

Interessanterweise hat die Wahl des Systemmodells Auswirkungen auf die Entdeckung möglicher Fehler. Zum Beispiel kann man in asynchronen Systemen nicht entscheiden, ob ein Rechnerknoten abgestürzt ist oder nicht [12]. Man benötigt bestimmte Zeitannahmen, um dies sicher entscheiden zu können. Diese Zeitannahmen können global über synchronisierte Uhren oder lokal durch sogenannte *timeouts* implementiert werden [21]. Wenn eine bestimmte Zeitdauer (der *timeout*) verstrichen ist ohne daß ein Rechner x eine Nachricht von einem Rechner y erhalten hat, so nimmt x an, daß y abgestürzt ist. Implementierungen von *timeout*-basierten Fehlerdetektoren reagieren in der Regel sehr empfindlich auf die Verletzung angenommener Zeitschranken, was zu unzuverlässigen Fehlerdetektoren führen kann [6], die ggf. ihre internen Zeitschranken dynamisch adaptieren.

Methoden der Fehlerdiagnose. Bei der Fehlerdiagnose in verteilten Systemen geht man davon aus, daß sich die einzelnen Rechnerknoten mittels spezieller Sensoren gegenseitig überwachen. Die Meßergebnisse werden an bestimmten Zeitpunkten zu einem sog. *Syndrom* (engl. *syndrome*) zusammengefaßt. Fehler in Rechnerknoten oder Sensoren erzeugen spezielle Syndrome, mit deren Hilfe diese Fehler diagnostiziert werden können [3].

4 Was können beide Gebiete voneinander lernen und was nicht?

Vergleich der Systemannahmen. Beiden Disziplinen liegt im Kern ein automatenbasiertes (d.h. zeitdiskretes) Systemmodell zugrunde, wobei der Zustand und die Zustandsübergänge in der Automatisierungstechnik eher an kontinuierlichen Funktionen orientiert sind. Das ergibt sich aus dem Vorhandensein der "physikalischen Komponente", die den verteilten Systemen üblicherweise fehlt. Außerdem sind Systeme in der Automatisierungstechnik durch die Notwendigkeit einer regelmäßigen Abtastung *immer* synchron im Sinne der Informatik (die obere Schranke der Abtastperiode ergibt sich aus dem Shannonschen Abtasttheorem). Jedoch reagieren die Systeme in beiden Bereichen empfindlich auf die Verletzung von Zeitschranken.

Vergleich der Fehlermodellierung. Bei der Fehlermodellierung gehen beide Disziplinen ähnliche Wege: Zustandsübergänge in den verteilten Systemen kann man als (nichtdeterministische) Funktion ansehen, die durch Überlagerung einer "Fehlerfunktion" zum modellierten Fehlverhalten führt.

Vergleich der Fehlererkennungsmethoden. Beiden Disziplinen liegt das selbe Grundprinzip bei der Fehlererkennung zugrunde: man muß einen "guten" von einem "schlechten" Zustand unterscheiden. Bei den heuristischen Methoden in der Automatisierungstechnik kommt die Grundlage für diese Unterscheidung von außen, d.h. durch die Codierung von Bedienerwissen. Uns sind keine Ansätze aus der Informatik bekannt, die ähnliche Ideen verfolgen, um beispielsweise aus gesammelten Informationen über Ausfallhäufigkeit, Last oder Kommunikationsverhalten entfernter Rechner die Zuverlässigkeit von Fehlerdetektoren zu erhöhen. Derartige Methoden werden aber immer anwendungsspezifisch bleiben und erfordern unter Umständen die Einbringung von KI-Techniken. Hier ist unserer Meinung nach noch Potential für weitere Forschungsvorhaben vorhanden.

Grenzwertüberwachung, signalbasierte Methoden und viele modellbasierte Ansätze aus der Automatisierungstechnik haben nicht nur im Hinblick auf ihre manchmal unsichere Aussagekraft deutliche Ähnlichkeiten zu der *timeout*-basierten Fehlererkennung in den verteilten Systemen. Daß letztere Methoden jedoch Aufgrund des *Fehlens* bestimmter Informationen entscheiden, ist nur auf den ersten Blick ein Gegensatz: Die Unterscheidung zwischen "guten" und "schlechten" Zuständen ist in beiden Fällen unsicher; die Entscheidungsgrundlagen können dynamisch adaptiert werden. Wenn jedoch in den verteilten Systemen eine Zustandsinformation (z.B. in Form einer eingegangenen Nachricht) vorliegt, dann kann man in der Regel entscheiden, ob ein Fehler aus der Fehlerannahme eingetreten ist. Dies ist in den angesprochenen Verfahren der Automatisierungstechnik aufgrund der unvollständigen Modellierung des physikalischen Systems nicht möglich.

Modellgestützte Verfahren, welche die physikalischen Eigenschaften des Systems *exakt* nachbilden (und darum eine genaue Fehlererkennung und -diagnose ermöglichen) scheinen in den verteilten Systemen keinen Sinn zu machen: durch das Fehlen der physikalischen

Komponente ist dort das System gleich dem Modell! Eine vollständige lokale Simulation eines entfernten Rechners zum Zweck der Fehlererkennung widerspricht auf den ersten Blick der Verteiltheitsannahme. Jedoch sind in der Informatik Verfahren üblich, welche die Prozesse eines Rechners entfernt *replizieren*: ein Replikat ist eine Kopie, welche dieselben Aufgaben übernimmt wie das Original. Durch den Vergleich beispielsweise der Ergebnisse einer Rechenoperation (allgemein: der Zustände) kann man auf das Vorhandensein und die Art eines Fehlers schließen. Uns sind jedoch keine Ansätze bekannt, die eine derartige Fehlererkennung mit Hilfe reduzierter Prozeßmodelle erreichen, d.h. durch die (u.U. lokale) Ausführung eines Moduls, welches nur die relevanten Aspekte des entfernten Prozesses wiederspiegelt.

5 Fazit und Ausblick

Wir haben angedeutet, in welcher Weise sich die Methoden der Fehlererkennung in der Automatisierungstechnik und den verteilten Systemen voneinander Unterscheiden und wo sie sich ähneln. Natürlich konnte der Vergleich beider Gebiete im Rahmen von sechs Seiten nur in aller Kürze geschehen. Neben Präzisierung der hier beschriebenen Sachverhalte muß der genaue Vergleich auch noch auf das Gebiet der Fehlerdiagnose ausgedehnt werden. Trotzdem haben sich als Ergebnis dieses Vergleiches mehrere Themenbereiche abgezeichnet, in denen noch Potential für weitere Forschungen steckt.

Literatur

[1] Y. Afek and S. Dolev. Local stabilizer. In *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing (PODC97)*, Seiten 287ff, 1997.

[2] A. Arora and S. Kulkarni. Detectors and correctors: A theory of fault-tolerance components. In *Proceedings of the 18th IEEE International Conference on Distributed Computing Systems (ICDCS98)*, Mai 1998.

[3] M. Barborak, A. Dahbura, and M. Malek. The consensus problem in fault-tolerant computing. *ACM Computing Surveys*, 25(2):171–220, Juni 1993.

[4] J. Beauquier, S. Delaët, S. Dolev, and S. Tixeuil. Transient fault detectors. In *Proceedings of the 12th International Symposium on DIStributed Computing (DISC'98)*, Lecture Notes in Computer Science 1499, Seiten 62–74, Andros, Griechenland, 1998.

[5] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon-Press, Oxford, 1995.

[6] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, März 1996.

[7] K. M. Chandy and L. Lamport. Distributed snapshots: determining global states of distributed systems. *ACM Transations on Computing Systems*, 3(1):63–75, 1985.

[8] C. M. Chase and V. K. Garg. Detection of global predicates: Techniques and their limitations. *Distributed Computing*, 11(4):191–201, 1998.

[9] K. Echtle. Fehlermodellierung bei Simulation und Verifikation von Fehlertoleranz-Algorithmen für Verteilte Systeme. In F. Belli, S. Pfleger, and M. Seifert (Hrsg.), *Software-Fehlertoleranz und -Zuverlässigkeit*, Informatik-Fachberichte 83, Seiten 73–88, 1984.

[10] K. Echtle. *Fehlertoleranzverfahren.* Springer, 1990.

[11] K. Echtle and J. G. Silva. Fehlerinjektion – ein Mittel zur Bewertung der Maßnahmen gegen Fehler in komplexen Rechnersystemen. *Informatik Spektrum*, 21(6):328–336, Dez. 1998.

[12] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, Apr. 1985.

[13] B. Freyermuth. *Wissensbasierte Fehlerdiagnose am Beispiel eines Industrieroboters.* Fortschr.-Ber. VDI 315, Reihe 8. VDI-Verlag, Düsseldorf, 1993. Dissertation TH Darmstadt.

[14] F. C. Gärtner. Fundamentals of fault tolerant distributed computing in asynchronous environments. Technischer Bericht TUD-BS-1998-02, TU Darmstadt, Juli 1998. Erscheint in *ACM Computing Surveys*, 31(1), März 1999.

[15] J. Gertler. *Fault Detection and Diagnosis in Engineering Systems.* Marcel Dekker, New York, 1998.

[16] T. Höfling. *Methoden zur Fehlererkennung mit Parameterschätzung und Paritätsgeleichungen.* Dissertation, TH Darmstadt, 1996.

[17] R. Isermann. *Identifikation dynamischer Systeme.* Springer, Berlin, 1992.

[18] R. Isermann (Hrsg.). *Überwachung und Fehlerdiagnose — Moderne Methoden und ihre Anwendungen bei technischen Systemen.* VDI-Verlag, Düsseldorf, 1994.

[19] R. Isermann. On fuzzy logic applications for automatic control, supervision and fault diagnosis. In *Proceedings of the Third European Congress on Intelligent Techniques and Soft Computing (EU-FIT)*, Seiten 738–753, Aachen, 1995.

[20] R. Isermann. Modellgestützte Überwachung und Fehlerdiagnose technischer Systeme. *Automatisierungstechnische Praxis*, 38:9–20, 48–57, 1996.

[21] J. Kaiser and E. Nett. Echtzeitverhalten in dynamischen, verteilten Systemen. *Informatik Spektrum*, 21(6):356–365, Dez. 1998.

[22] T. Knuth. Schadenfrüherkennung durch Schwingungsanalysen — Neue Möglichkeiten in der Instandhaltung. *Der Maschinenschaden*, 61:70–74, 1988.

[23] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, Juli 1982.

[24] O. Nelles, S. Ernst, and R. Isermann. Neuronale Netze zur Identifikation nichtlinearer dynamischer Systeme: ein Überblick. *Automatisierungstechnik*, 45(6):251–262, 1997.

[25] H. Niemann. *Methoden der Mustererkennung.* Akademische Verlagsgesellschaft, Frankfurt, 1974.

[26] F. B. Schneider. What good are models and what models are good? In S. Mullender (Hrsg.), *Distributed Systems*, Kapitel 2, Seiten 17–26. Addison-Wesley, zweite Auflage, 1993.

[27] H. W. Schüßler. *Digitale Signalverarbeitung.* Springer, Berlin, 1994.

[28] R. Schwarz and F. Mattern. Detecting causal relationships in distributed computations: in search of the holy grail. *Distributed Computing*, 7:149–174, 1994.